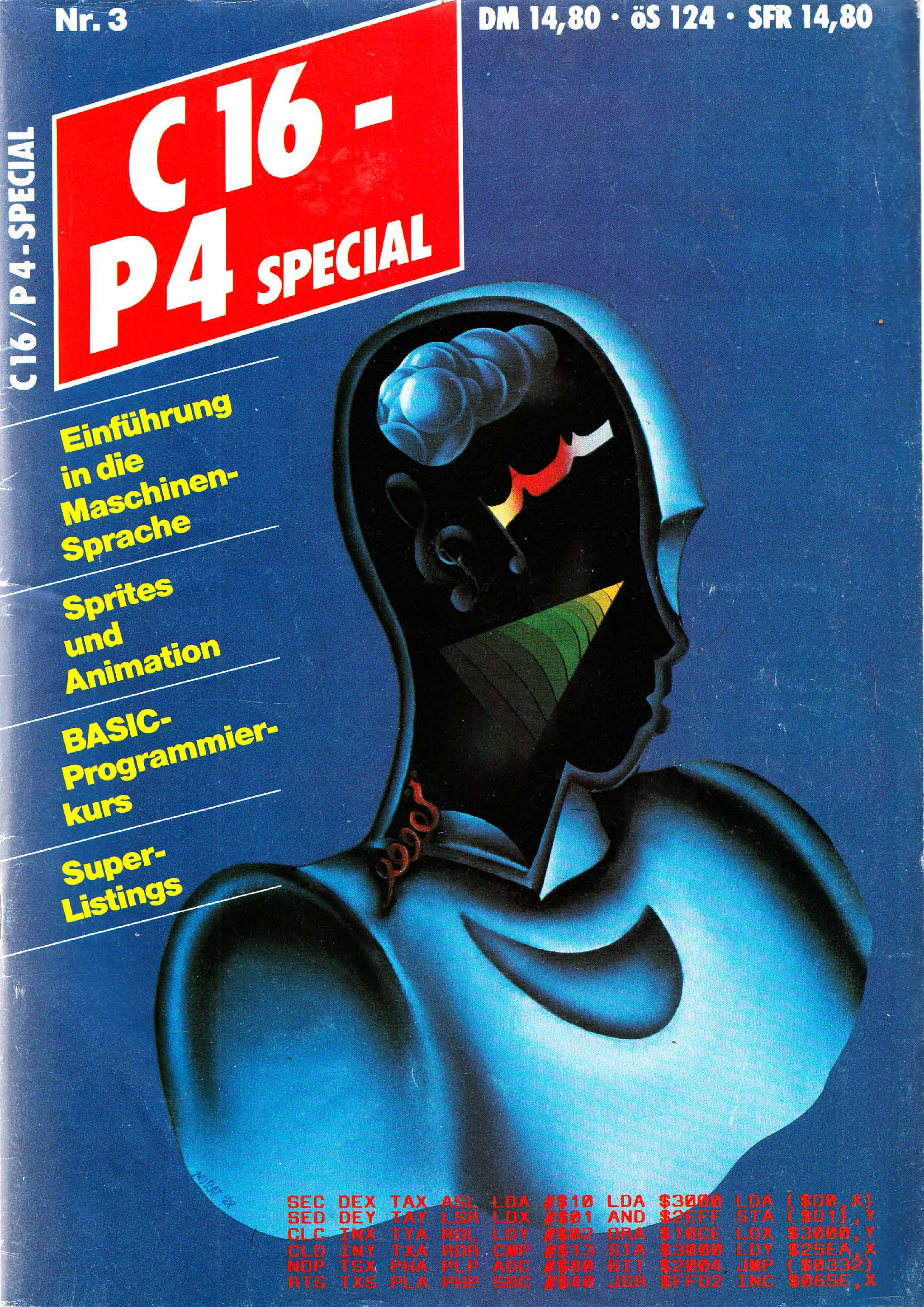


Nr. 3
DM 14,80 · öS 124 · SFR 14,80
C 16/P 4-SPECIAL
C 16 -
P 4 SPECIAL
Einführung in die Maschinen-Sprache
Sprites und Animation
BASIC-Programmier-kurs
Super-Listings
SEC DEX TAX ASL LDA #$10 LDA $3000 LDA ($D0,X)
SED DEY TAY LSR LDX #$01 AND $2EFF STA ($D1),Y
CLC INX TYA ROL LDY #$02 ORA $10CE LDX $3000,Y
CLD INY TXA ROR CMP #$13 STA $3000 LDY $25EA,X
NOP TSX PHA PLP ADC #$80 BIT $2004 JMP ($0332)
RTS TXS PLA PHP SBC #$40 JSR $FFD2 INC $065E,X

EDITORIAL

Unser neues C16-P4-SPECIAL wartet mit Glanzpunkten auf, die für viele vielleicht ein absolutes Novum bedeuten. Wußten Sie schon, daß die C16/116/Plus4-Rechner, obwohl kein spezieller Grafikchip sie unterstützt, dennoch mit Sprites aufwarten können? Mit Sprites meinen wir Objekte, die als vollständige Einheit über den Bildschirm bewegt werden können. Die Bewegung dabei hat nicht sprunghaft um jeweils ein Zeichen zu erfolgen, sondern koninuierlich um jeweils ein Pixel, oder anders ausgedrückt, um jeweils ein Achtel-Zeichen. Daß dieses möglich sein muß, sehen wir an professioneller Spielesoftware, die es für viel Geld zu kaufen gibt. Selbst solche Sprites zu erzeugen, womöglich mit ein paar einfachen BASIC-Befehlen, war bisher leider nicht machbar. Jetzt aber können auch Sie richtige Bewegung in das Spiel bringen. Die BASIC-Erweiterung S&A-BASIC, die wir in diesem Heft für alle C16/116/Plus4-Rechner bringen, die auf 64 KByte erweitert sind, ist speziell für Sprites und Animation gedacht, damit Sie sich nicht mehr sagen lassen müssen, Ihr Rechner hat ja nicht einmal Sprites. Zeigen Sie einem solchen Skeptiker, was Ihr Rechner tatsächlich auf dem Kasten hat.

Viele, die einen Rechner haben, haben meist auch Grundkenntnisse in BASIC. Doch BASIC kennen heißt noch lange nicht, auch eigene Anwendungen in BASIC selbst programmieren können. Eine neue Artikelreihe mit dem Titel „So programmiere ich in BASIC" weist Ihnen den Weg. In diesem Heft wird diskutiert, was BASIC bringt. Vor- und Nachteile werden aufgezeigt. Nachdem BASIC sich als recht gut geeignet erweist, auf die Schnelle ein Programm zu verfassen, zeigt der Autor, wie er an ein zu erstellendes Programm herangeht, wie ein Programm geplant, und in welchen Schritten es realisiert wird. Er demonstriert es zuletzt an einem Programmgerüst, das durch Ergänzungen von Daten zu einem individuellen Quizprogramm gemacht werden kann. Auch wer sagt, programmieren könne er schon, wird sicherlich vom Programmstil her von dieser Serie profitieren können. Tips und Tricks zum Programmieren finden sich darin reichlich.

Ziemlich breiten Raum haben wir in diesem C16-P4-SPECIAL der Einführung in Maschinensprache gewidmet. Salamitaktik zu praktizieren und den Befehlssatz scheibchenweise zu servieren, alle zwei Monate ein bißchen etwas, erschien nicht als der angemessene Weg. Wer sich ein Auto kaufen will, um damit zu fahren, kauft sich ja auch nicht zuerst die Reifen, dann das Lenkrad und die Gangschaltung und später irgendwann einmal auch den Rest. Er wird daran nicht viel Freude haben, denn fahren kann er damit nicht. Wenn er alles beisammen hat, sind die Teile, die er zuerst erworben hat, vielleicht schon wieder verrostet, in unserem Falle eben vergessen. Es besteht ein Unterschied zwischen dem Lesen von Büchern über das Autofahren und dem Autofahren selbst. Dies trifft auch zu auf die Befehle der CPU und den Umgang mit ihnen. Die Einführung wird begleitet von einem Programm, durch welches die Befehle der CPU direkt eingegeben und deren Wirkungen beobachtet werden können. So können Sie die Register des TED-Chips und das Port-Register manipulieren, worauf sich Änderungen ergeben, die Sie entweder direkt sehen oder hören, die Sie am seriellen Port messen, oder an geänderten Registerinhalten wahrnehmen können. Durch Aufrufen der Standard-Ein- und -Ausgaberoutinen können direkt Daten auf die verschiedenen Geräte übertragen werden. Mit dem Maschinenmonitor, den die C16/116/Plus4-Rechner dem C64 voraushaben, erstellen Sie Ihre ersten kleinen Maschinenroutinen. Wichtig ist diese Einführung besonders, weil wir Ihnen in den weiteren Heften nicht nur Tips und Tricks und Utility-Programme zu Ihrem Rechner geben möchten, sondern Ihnen diese auch gerne erklären würden. Letzteres geht aber ohne die erforderlichen Grundlagen nicht.

Von vielen herbeigesehnt wurde eine Textverarbeitung mit 80-Zeichen-Darstellung auf dem Bildschirm. Wir haben Sie getestet. Einen ausführlichen Bericht, der nicht nur irgendein Urteil abgibt, sondern umfassend über die Möglichkeiten von PLUS-TEXT-80 informiert, finden Sie in diesem Heft.

Den Freunden von Hardware-Basteleien stellen wir das Messen mit dem Joystickport vor. Durch das Abfragen eines TED-Registers mit einer Maschinenroutine können die einzelnen Pins der Joystickports abgefragt werden. Da ergeben sich sicherlich eine Menge von Anwendungsmöglichkeiten. Eine davon, den Bau einer einfachen Alarmanlage, finden Sie im Heft.

Ob für den Anwender, den Hardware-Bastler, für den, der seine Maschine besser kennenlernen will, für jedermann haben wir stets Interessantes und Nützliches parat. Sollten Sie finden, daß einem gewissen Gebiet zuwenig Aufmerksamkeit geschenkt wird, so schreiben Sie uns oder nehmen unsere Hotline am Mittwoch von 15 bis 19 Uhr in Anspruch. Wir sind für Anregungen stets dankbar.

Ihr
COMMODORE WELT-Team

INHALT

TEST & TECHNIK

PLUS-TEXT 80
80 Zeichen auf dem Bildschirm — Seite 16

REPORT

Der letzte Kaiser kommt um sechs
Computereinsatz in der Videothek — Seite 4

Aug' in Auge
Neue Monitortechnologie bringt Entlastung für die Augen — Seite 7

GRUNDLAGEN

Rechnen in anderen Zahlensystemen
Rechnen mit Binärzahlen — Seite 11

Das Hexadezimalsystem — Seite 14

Black Beauty
Komplette Einführung in die Maschinensprache — Seite 56

SERIE & SERVICE

So programmiere ich in BASIC
Neue Serie zeigt, wie BASIC sich für die persönlichen Belange nutzen läßt — Seite 138

Sprachgewirr wie in Babylon
Programmiersprachen zur Auswahl, Teil 3: OCCAM bis V.I.P. — Seite 18

Checksummer
Kontrollierte Eingabe — Seite 46

TIPS & TRICKS

Vom Joystick bis Script/Plus
Alarmanlage mit Joystickports — Silbentrennung mit Script/Plus — Seite 32

Hätten Sie es gewußt?
Kleine Hilfen für den täglichen Kampf mit dem Computer — Seite 31

Und sie passen doch
Stecker und ihr richtiger Anschluß — Seite 129

Karteikasten
Tips & Tricks für Ihren Rechner — Seite 135

Dialog
An uns, mit uns — Seite 29

LISTINGS

Sprites und Animation
BASIC-Erweiterung bringt Bewegung ins Spiel — Seite 36

Zeichencreator
Eigene Zeichen entwickeln — Seite 43

Kalender 2300
Kalenderrechnung bis zum Jahre 2299. Grundlagen und Programm — Seite 48

Seq-Files
Lesen von ASCII-Texten — Seite 95

Paint-System
Malprogramm mit grafischer Benutzeroberfläche — Seite 96

Feste Ausgaben
Überblick über regelmäßig anfallende Kosten — Seite 102

Haste Töne?
Der Computer spielt nach Noten — Seite 109

Startmenü
Programmanwahl per Tastendruck — Seite 115

Farbdemo
Optimale Monitoreinstellung — Seite 117

Disk-To-Tape
Sicheres Kassettenkopieren — Seite 128

CPU-Trainer
CPU-Befehle zum direkten Eingeben — Seite 93

DATA-Wandler
Maschinenprogramme in Datazeilen ablegen — Seite 94

Silbentrennung
Zur Bearbeitung von Script/Plus-Texten — Seite 34

Schalter.C16
Abfrage optischer Schalter am Joystickport — Seite 35

SPIELE

Gorgul
Brechen Sie die Herrschaft des Tyrannen — Seite 118

3D-Design-Golf
Trotz HiRes-Grafik ohne Erweiterung lauffähig — Seite 122

Family-Quiz
Quiz-Programm zum individuellen Ausbau — Seite 142

STÄNDIGE RUBRIKEN

Börse
Für jeden etwas — Seite 134

Editorial — Seite 2

Impressum — Seite 28

LOAD & RUN

Das Spiele-Magazin
Das aktuelle Spielemagazin finden Sie ab Seite 65

Der PC in der Videothek

DER LETZTE KAISER KOMMT UM SECHS

Filmverleih leicht gemacht – durch Computereinsatz

Was macht der Besitzer einer Videothek am Sonntag? Er brütet über Statistik und Monatsabrechnung. Ein anderer kann künftig spazierengehen: Er hat sich einen Computer zugelegt.



„Ich hätte bereits vor zwei Jahren anfangen sollen, mit dem Computer zu arbeiten." Günther Tauer, Chef der Karlsfelder Videothek „Video-Kiste", hat lange genug an Sonntagen über Statistiken gebrütet. Der Wille zur Umstellung war wohl vorhanden – allein, es hat das geeignete Programm gefehlt.

Dieser Mangel ist seit Februar behoben. Ein befreundeter Programmierer hat ihm Vidiplus entwickelt, das auf die Bedürfnisse seines Geschäfts abgestimmt ist. Warum hat er sich nicht für eines der Fertig-Programme entschieden, die es bereits in allen Variationen und Preisklassen gibt?

Tauer winkt ab: „Ich habe mir das eine oder andere angeschaut. Ist zwar alles gut und schön, aber wenn ich Änderungen brauche, kostet das zusätzlich. Mit hundert Mark und mehr ist man gleich dabei."

Die täglichen Arbeiten lassen sich mit Vidiplus in sechs Masken abwickeln: Verleihgeschäft, Kundendatei, Filmeingabe, Bar-Code-Druck, Datensicherung, Mahnwesen.

Im Moment wird lediglich mit der Kundendatei und der Filmeingabe gearbeitet. Die weiteren Programmteile sollen in Kürze installiert werden. Problemlos zu handhaben sei die Sache auf jeden Fall, meint Tauer.

„... eine Riesenerleichterung"

„Was mir Roland Haas, der Programmierer, beibringt, werde ich vereinfacht ans Personal weitergeben." Etwa eine Stunde hat er dafür veranschlagt, zumal durch den Bar-Code-Lesestift einiges an Unterweisungszeit gespart würde.

Für den Videotheken-Besitzer ist das Programm „eine Riesenerleichterung". Obgleich die Video-Kiste zu den kleineren Geschäften dieser Art zählt, bietet sie auf etwa 60 Quadratmetern Platz für immerhin 3000 Filme. Rund 2500 verschiedene Titel sind derzeit im Angebot. Durchschnittlich 100 werden an den Werktagen verliehen, samstags sind es gar um die 300.

„An Stoßzeiten standen wir manchmal zu dritt hinter der Theke", sagt Tauer, der neben seiner Frau noch zwei Aushilfen für die Vormittage beschäftigt. „Manchmal war es das reine Chaos: Vom Kunden mußte ich außer dem Namen auch wissen, wann er das letzte Mal da war. Konnte er sich nicht genau daran erinnern, begann die ewig lange Sucherei in der alten und neuen Karte." Was noch dazu kam: Verleih-Formular ausfüllen, auf die unterschiedlichen Preise achten, kassieren.

Das geht jetzt alles schneller: Der Kunde ist mit seinen Daten gespeichert, ebenso die Filme. Der Rest ist

Schluß mit den Strichlisten

ein Kinderspiel: Eingabe der Kundennummer, Namensvergleich, Heraussuchen des gewünschten Films. Preis und Titel werden über Bar-Code eingelesen. Weitere Arbeitsersparnis wird es geben, wenn die neuen lesbaren Kundenkarten verteilt sind. Tauer sieht darin auch ein psychologisches Moment: „Die Dinger werden bestimmt nicht einfach zu Hause vergessen. Die Leute schätzen diese technischen Spielereien. Die mögen es ganz gern, wenn etwas piepst."

Nicht nur fürs tägliche Verleihgeschäft läßt sich das Programm nützen. Schließlich will ein „Videothekar" auch wissen, wie häufig welche Filme über den Ladentisch gehen. Bisher hat Tauer Strichlisten geführt: „Ich muß einen Überblick haben, ob die verschiedenen Titel ihren Einkaufs-

Von links nach rechts; Günther Tauer und Roland Haas führen unserer Reporterin Waltraut Schmidt das System vor.

Unten: Durch den Einsatz der EDV kann sich G. Tauer intensiver den Wünschen seiner Kunden widmen.



preis auch eingespielt haben." Diesen Überblick hat er jetzt auf dem Bildschirm. Ebenso läßt sich daraus ersehen, mit welchen Filmherstellern welcher Umsatz erzielt worden ist.
„Von den faulen Kunden wird mir jetzt auch keiner mehr durch die Lappen

Faulen Kunden keine Chance

gehen." Tauer spricht damit ein Problem der meisten Videotheken an: Da leiht sich einer Kassetten aus und verschwindet damit auf Nimmerwiedersehen.
Zwar werden Listen mit den Namen dieser schwarzen Schafe verteilt. In der Kundenkartei werden sie mit einem Sperrvermerk versehen. Nur: „Wenn einer behauptet, er sei zum ersten Mal hier, habe ich das Nachsehen." Diese Zeitgenossen werden jetzt unter einer bestimmten Nummer gespeichert. Da sich jeder neue Kunde sowieso zunächst ausweisen muß, genügt ein Blick auf den Bildschirm: ertappt.
Das Programm zeigt auch an, wenn die Leihfrist überschritten wurde. Etwa 150 Mark, schätzt Tauer, gehen pro Jahr an Nachgebühren verloren – in der Hauptsache durch Unachtsamkeit, wenn viel Betrieb ist. Kein allzu großer Verlust zwar, doch wenn er sich vermeiden läßt ... „Die Leute müssen jetzt warten, bis alles eingelesen und überprüft ist."
Mitunter werden beschädigte Videos zurückgegeben. Wer dafür verantwortlich war, läßt sich nur schwer feststellen. Eine Hilfe bietet Vidiplus: Die letzten drei Kunden, die den Film geliehen hatten, werden aufgelistet.



Links:
Unauffällig und platzsparend paßt sich das System in das Arbeitsfeld ein.

Unten:
Eine der Bildschirmmasken zeigt alle Informationen über die angebotenen Filme. Die Daten der Kunden sind in einer ähnlichen Maske enthalten.

Alles ist jedoch noch nicht machbar: Eine Kundin möchte „Der letzte Kaiser" ausleihen. Tauer schaut auf eine Liste: „Tut mir leid, nicht vor nächster

Reservierung fehlt noch

Woche. Höchstens noch heute abend, ab 18 Uhr." Die Reservierung ist im Programm nicht enthalten, es wird noch immer mit der bewährten Listen- und Zettel-Methode gearbeitet. Für die Zukunft ist aber auch sie vorgesehen.
Der Videotheken-Besitzer liebäugelt noch mit einer Textverarbeitung zum Erstellen von Serienbriefen. Ein Drucker ist ja bereits vorhanden.
Steht aber der finanzielle Aufwand in einem akzeptablen Verhältnis zum Ertrag? Bereits die Programme, die derzeit angeboten werden, kosten, je nach Umfang, zwischen 2.000 und 10.000 Mark. Wie teuer war der ganze Spaß? Rund 9.500 Mark insgesamt – für Computer, Drucker und Programm, sagt Haas. Mit Textverarbeitung und Bar-Code würden die Kosten auf etwa 12.000 Mark klettern, rechnet sich Tauer aus. Er ist mit weiteren Händlern in einer Einkaufs-Cooperative zusammengeschlossen, das Programm können alle gemeinsam nutzen.
„Klar, für einen Laden mit etwa fünf- bis sechshundert Titeln lohnt sich ein Computer nicht." Das habe auch mit

„Ohne Computer geht nichts mehr"

der Kundenfrequenz zu tun. Aber ab anderthalbtausend Filmen und Kunden lasse sich darüber reden.
Für ihn hat die Sache auf jeden Fall Hand und Fuß. Da sich die täglich anfallenden Arbeiten einfacher und schneller erledigen lassen, kann er in Stoßzeiten auf einen Mitarbeiter verzichten. Überdies: „Ich kann jederzeit nachprüfen, mit welchem Filmgenre und welchen Firmen ich keinen Umsatz mache." Durch gesteuerten Einkauf kann er damit eine Menge Geld einsparen. Für ihn steht fest: „Ohne Computer geht nichts mehr."

was



Neue Monitortechnologie

Auge in Auge

Gewerkschaften und Betriebsärzte streiten sich schon lange mit der Arbeitgeberseite über das Für und Wider von Bildschirmarbeitsplätzen. Sie seien gesundheitsschädlich und nicht zumutbar, argumentieren die einen. Alternativen sind nach Ansicht der Arbeitgeber nicht abzusehen. Auch die Hersteller von Monitoren setzen sich mit der Problematik auseinander.

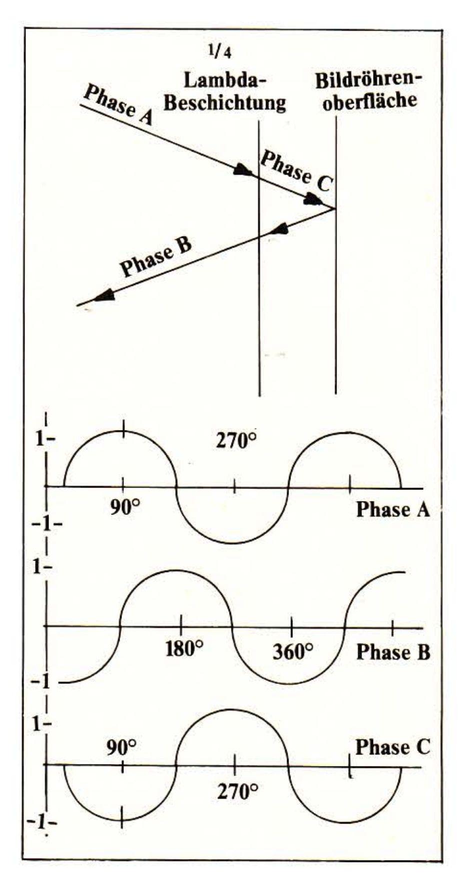
Die meisten Firmen, die derzeit versuchen, sich im Markt der hochwertigen Peripheriegeräte zu etablieren, orientieren sich an bereits existierenden Produkten der Konkurrenz.
Anders die Marketing-Linie der belgischen Firma Etap. Ihre Devise heißt: Erst Grundlagenforschung betreiben und dann produzieren. Resultat ist eine Reihe von Erkenntnissen, die jeden, der an einem Bildschirm arbeitet oder solche Arbeitsplätze plant, interessieren sollten.
Personen, die täglich vier bis fünf Stunden, an 200 Arbeitstagen, und das 30 Jahre lang an einem Bildschirm arbeiten, verbringen, hochgerechnet 24.000 Stunden ihres Arbeitslebens vor ihrem Monitor. Wenn es durch Verbesserung der Arbeitsbedingungen gelingen sollte, eine Steigerung der Produktivität dieser Personen von zehn Prozent zu erzielen, so ist dies ein Fortschritt.
Die Ergebnisse der Grundlagenforschung von Etap zeigen eine denkbare Steigerung der reinen Arbeitsgeschwindigkeit des Benutzers um 25 Prozent. Hierzu kommen noch derzeit schlecht meßbare Verbesserungen bei der Verhütung von Augenschäden und die Steigerung des allgemeinen Wohlbefindens von Personen, die am Bildschirm arbeiten müssen.

Das Auge im Detail

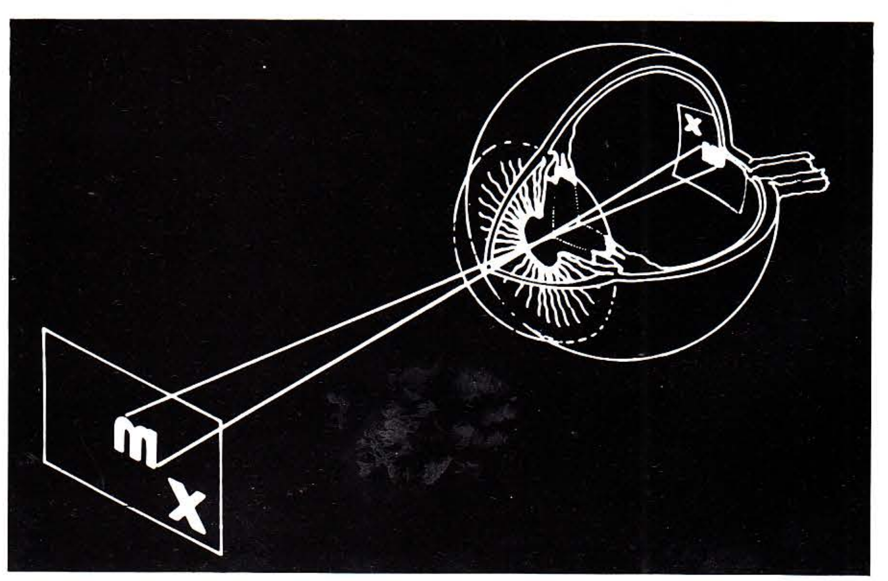
Der Monitor eines Computers und das menschliche Auge sind zwei Komponenten, die auf einander abgestimmt werden müssen. Dies ist nur möglich, wenn beider Aufbau und Funktion bis ins Detail bekannt ist.
Die wichtigsten Teile des menschlichen Auges sind die Linse, die Pupille und die Netzhaut (Retina). Die Linse projiziert eine Abbildung auf die Netzhaut und sorgt für die Schärfe dieser Abbildung. Da unsere Augen es gewohnt sind, immer scharf zu sehen, versuchen unsere Augen beim Lesen einer unscharfen Vorlage ständig durch Strecken und Dehnen von Muskeln die Linse so einzustellen, daß eine scharfe Abbildung auf der Netzhaut entsteht. Dies ist bei Verwendung einer unscharfen Vorlage ein hoffnungsloses Unterfangen. Unsere Augen ermüden durch das ständige Nachregulieren sehr schnell. Wird trotzdem weiter gearbeitet, läßt die Konzentration nach, und die Fehlerhäufigkeit des Arbeitenden steigt stark an. Andere Beschwerden, wie zum Beispiel Kopfschmerzen, folgen.
Jeder, der sich schon einmal mit Fotografie beschäftigt hat, kennt den Begriff Schärfentiefe. Gemeint ist eine Abhängigkeit zwischen Blendenöffnung und der Tiefe des Bereichs, der scharf dargestellt werden kann.
Die Pupille übernimmt im menschlichen Auge die Funktion der verstellbaren Blende eines Foto-Objektives. Je nach Lichteinfall öffnet oder schließt sich die Pupille. Bezogen auf die Arbeit am Bildschirm heißt dies: Ist der Bildschirm sehr hell, schließt sich die Pupille und eine große Schärfentiefe wird erreicht. Ist die Bildschirmdarstellung dunkel, so öffnet sich die Pupille und das Auge muß die Sehschärfe viel häufiger regulieren.

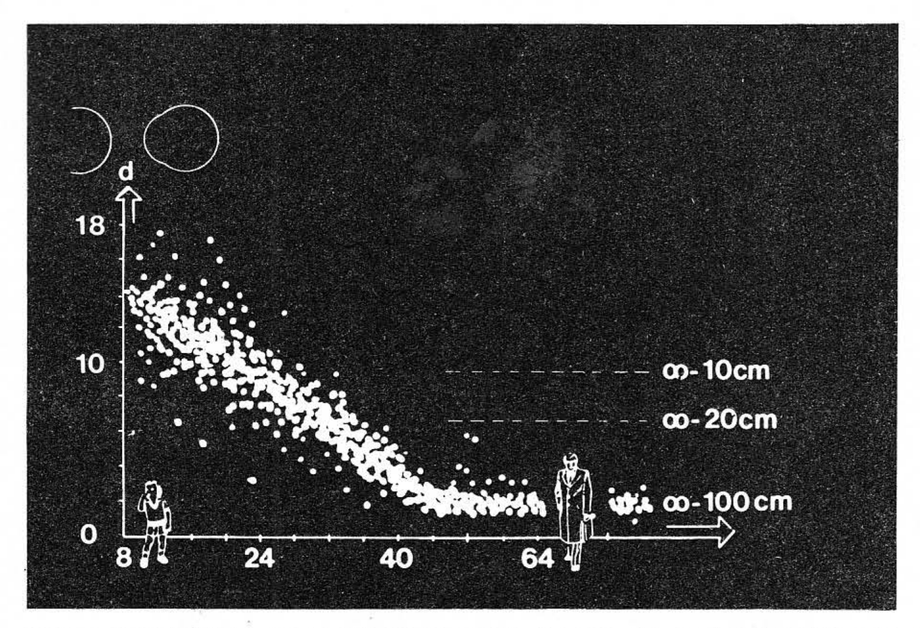
Ältere Augen brauchen mehr Licht

Personen, die ständig zwischen Papiervorlagen und Monitordarstellung wechseln, strapazieren bei großen Helligkeitsunterschieden ihre Augen erheblich mehr, als bei nahezu gleicher Helligkeit von Papier- und Bildschirmdarstellung. Ein weiterer Faktor ist die mit steigendem Alter abnehmende Empfindlichkeit unserer Augen. Je älter die Person ist, die an einem Bildschirm arbeitet, je mehr Licht muß ein Monitor abgeben, um der Person ein optimales Arbeiten zu ermöglichen. Eine andere Voraussetzung für ein zügiges Arbeiten an einem Monitor wird nicht im Alter, sondern in frühester Kindheit, in den ersten drei Lebensjahren, geschaffen. Zellen in unserem Gehirn die für das Sehen und Erkennen zuständig sind, werden in dieser Zeit „verdrahtet". Diverse geometrische Grundmuster werden in unseren Gehirnen als Hardwaremuster abgelegt. Alle Zei-

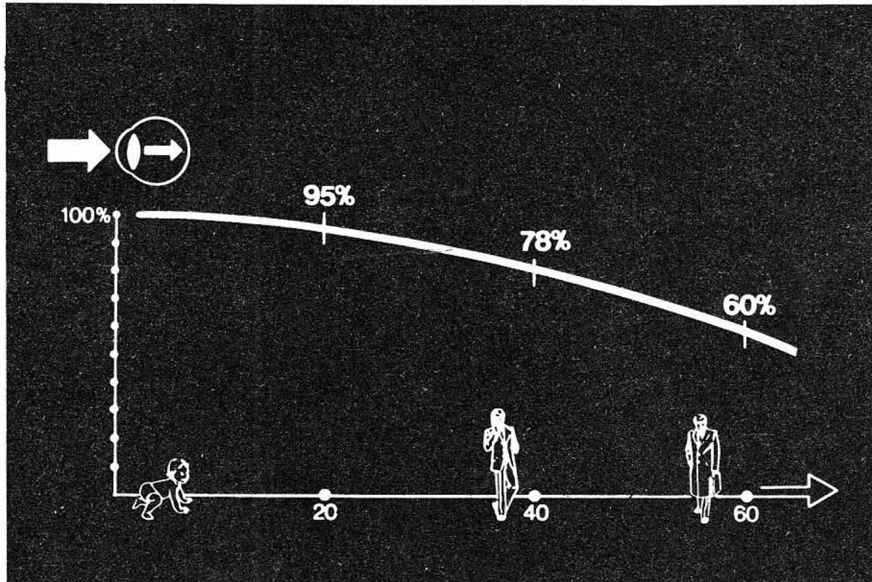


Eine in Phase A auf den Monitor treffende Welle wird bei Durchdringen der 1/4 Lambda-Beschichtung um 90 Grad-Phasen verschoben. Hierauf wird sie in Phase B an der Bildröhrenoberfläche reflektiert. Auf ihrem Weg nochmals um 90 Grad verschoben, ergibt eine Addition dieser Welle (Phase C) mit einer Eintreffenden den Wert 0.





Mit zunehmendem Alter nimmt die Fähigkeit des Nahsehens stark ab.



Die Lichtempfindlichkeit unserer Augen reduziert sich mit steigendem Alter bis auf rund 60 Prozent.

chen und Abbildungen, die nicht in dieser Ablage zu finden sind, muß unser Gehirn mit einer Auswerte-Software aus diesen Grundmustern konstruieren. Dies kostet viel mehr Zeit, als eine Auswertung mit Hilfe einer Hardware.

Die Anzahl und die Typen von Mustern, die unser Gehirn als Hardware ablegt, sind abhängig von der Umgebung, in der wir während der ersten drei Lebensjahre aufwachsen. Umgeben von Hochhäusern und Beton-Architektur bilden gerade Linien und rechte Winkel den Grundstock unserer Wahrnehmungen. Personen, die in freier Natur aufwachsen, übernehmen die dort vorkommende Formenvielfalt und verfügen über einen bedeutend umfangreicheren Vorrat an optischen „Hardwarelösungen".

Wird ein Text vom Bildschirm gelesen, so verhält es sich ähnlich. Von dem Arbeiten mit bedrucktem Papier sind wir es gewöhnt, durchgehende Linien zu erkennen und als Buchstaben zu interpretieren, welche wir dann zu Wörtern zusammensetzen.

Alle Monitore, die eine helle Schrift auf dunklem Grund darstellen, sind nicht in der Lage, kontinuierliche Linien abzubilden.

Deutlich erkennbar werden alle Zeichen aus Punkten zusammengesetzt. Unser Gehirn muß aus diesen Punkten mit Hilfe der „Software" ein Zeichen zusammensetzen. Monitore, die einen schwarzen Buchstaben auf weißem Grund darstellen, bilden kontinuierliche Linien ab. Diese Zeichen können im Gehirn mit Hilfe der „Hardware" ausgewertet werden. Dadurch wird der Lese- und Verständnisvorgang stark beschleunigt.

1974 an der Universität in Toronto/Canada, durchgeführte Testreihen zeigten, daß es bei der damaligen Monitortechnologie 30 Prozent mehr Zeit in Anspruch nahm, einen Text von einem Bildschirm zu lesen, als von einem Blatt Papier. Aktuelle, von Etap initiierte Untersuchungen an der Universität von Groningen zeigen, daß dieses Defizit durch Verbesserungen der Monitortechnologie vollständig ausgeglichen werden konnte. Erstaunlich ist, das ein von einem Monitor gelesener Text sogar schneller und besser verstanden wird, als ein von Papier gelesener.

Gestört wird der Nachbar

Bildschirme, die, bedingt durch eine zu geringe Zeilenfrequenz flimmern, stören nicht so sehr die Person, die direkt davor sitzt. Ein Flimmern ist bedeutend stärker und störender, wenn der Monitor aus spitzem Winkel betrachtet wird. Benachbarte Arbeitsplätze werden hierdurch beeinflußt.

Reflexionen auf einem Monitor sind ein jedem Anwender bekanntes Problem. Durch eine abgestimmte Raumbeleuchtung und eine vernünftige Aufstellung des Gerätes können sie gemindert werden. Die Technik der verwendeten Monitore muß das ihre dazu beitragen, Reflexionen auf ein Minimum zu reduzieren. Eine Möglichkeit bietet ein Ätzen der Bildröhrenoberfläche. Störende Reflexionen verschwinden hierdurch fast vollständig. Gleichzeitig beeinträchtigt das Ätzen aber die Schärfe und somit die Lesbarkeit der Abbildung deutlich.

Eine Beschichtung des Röhreninneren ist derzeit der beste Kompromiß zwischen Reflexion und Schärfe. Ein Viertel-Lambda-Beschichtung heißt eine der neuesten Technologien auf diesem Sektor.

Das bedeutet, daß Fremdlicht auf dem Weg zur Röhrenoberfläche, bedingt durch eine metallische Beschichtung des Röhreninneren, um ein Viertel seiner normalen Wellenlänge von einem Lambda verzögert wird. Reflektierte Strahlen müssen diese Verzögerung zweimal mitmachen und treffen so mit einer Verspätung von einer halben Wellenlänge wieder mit einfallen-

den Lichtstrahlen zusammen. Wird eine Welle, das ist ein Lichtstrahl physikalisch betrachtet, um die Hälfte ihrer Länge verzögert, so spricht man auch von einer Phasenverschiebung von 180 Grad. Das Mischen einer Welle mit einer Welle gleicher Frequenz, aber um 180 Grad gedrehter Phase, führt zu einer Auslöschung derselben. Somit können Reflexionen fast vollständig verhindert werden. Abgesehen davon sind Bildschirme mit weißem Untergrund und schwarzer Schrift sowieso bedeutend weniger anfällig für Reflexionen als solche mit schwarzem Untergrund.

Schlagwort Strahlenbelastung

Strahlenbelastung ist ein in jüngster Zeit oft gebrauchtes Wort. Auch die Bildröhren von Monitoren geben Strahlung ab. Sie entsteht durch die Beschleunigung von Elektronen mit großer Energie und hoher Frequenz. Auf jedem Monitor klebt schließlich ein Schildchen, auf dem die Abschirmung der Röhre dokumentiert ist. Die verbleibende Reststrahlung liegt deutlich unter der Strahlenabgabe anderer Haushalts- und Gebrauchsgegenstände. Die Strahlenbelastung eines Kühlschranks oder Staubsaugers ist, je nach Typ, bis zu zehn mal größer als die eines guten Monitors.

Die bisher gewonnenen Erkenntnisse führen zu einem Pflichtenheft, dem ein Monitor neuester Technologie entsprechen müßte. Resultierend aus Bedingungen, die unser Auge, unsere Arbeitsplatzumgebung und Gesetze der Physik stellen, wurde in den Laboratorien von Etap in Belgien eine Reihe von Ganzseitenmonitoren in den Formaten DIN A4 und A3 entwickelt, die den neuesten Stand der Technik widerspiegeln. Vorgesehen für den Einsatz an IBM-kompatiblen Personal-Computern und bei der MacIntosh-Familie sind sie in der Lage, die Standardmonitore zu ersetzen.

Wir haben die IBM-Version des DIN A4-Monitors Neftis getestet. Bestandteile des Testgerätes waren, nebendem Monitor, eine lange PC-Steckkarte, eine englische Bedienungs- und Installationsanleitung sowie zwei Disketten mit den erforderlichen Bilschirmtreibern.

Bereits nach Überfliegen der ersten Seiten des Handbuchs war klar, daß ein so komplexes Peripheriegerät nur von einem Fachmann installiert werden kann.

Nach zwei Anrufen beim Distributor gelang es uns, die Bildschirmkarte an den Schneider AT 2640 anzupassen. Hierzu müssen zuerst die DIP-Schalter gefunden werden, die die computereigene Bildschirmkarte abschalten. Danach kann der neue Bildschirmtreiber integriert werden. Erscheint beim Einschalten des Computers eine Meldung, die eine erfolgreiche Treiberinstallation quittiert, so steht unter DOS der gesamte, DIN A4 große, Bildschirm zur Verfügung. Ein Directory mit 60 Zeilen ist schon eine imposante Erscheinung.

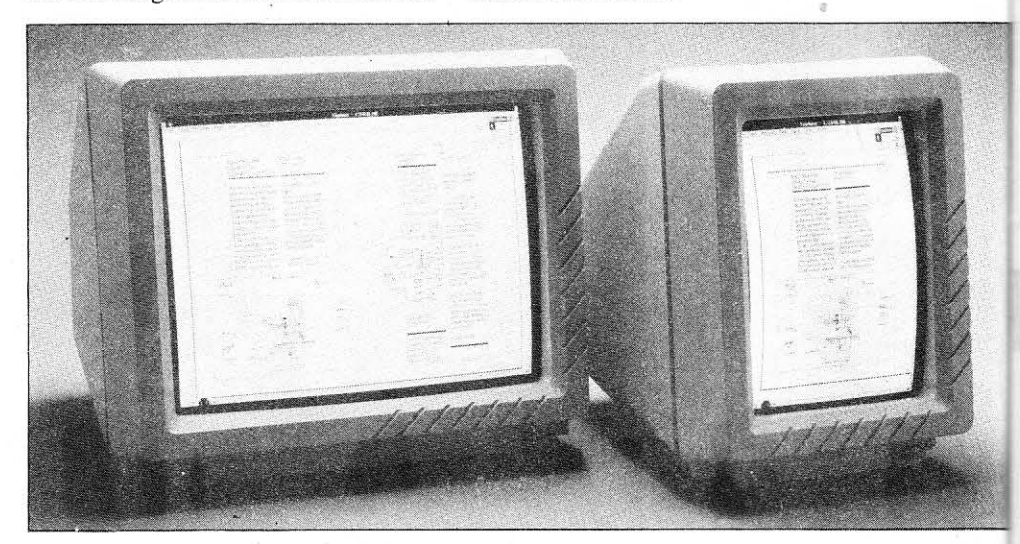
Die nächste Hürde, die es zu überwinden gilt, ist die Anpassung der Software. In einer großen Zahl von Standardanwendungen sind die Etap-Monitore im Installationsmenü bereits enthalten. Weitere Treiber befinden sich auf den mitgelieferten Disketten. Diese Treiber müssen mit einem Editor in die entsprechenden Installationsmenüs integriert werden.

Ausprobiert haben wir den Monitor mit den Benutzeroberflächen GEM und Windows. Auch Word und Wordstar können beispielsweise den kompletten Screen nutzen. Programme, die nicht den gesamten Bildschirm nutzen können, sind trotzdem lauffähig. Die Etap-Monitore emulieren CGA- und Herkulesmodus. So ist es zum Beispiel möglich, zwei Herkulesscreens übereinander und gleichzeitig darzustellen.

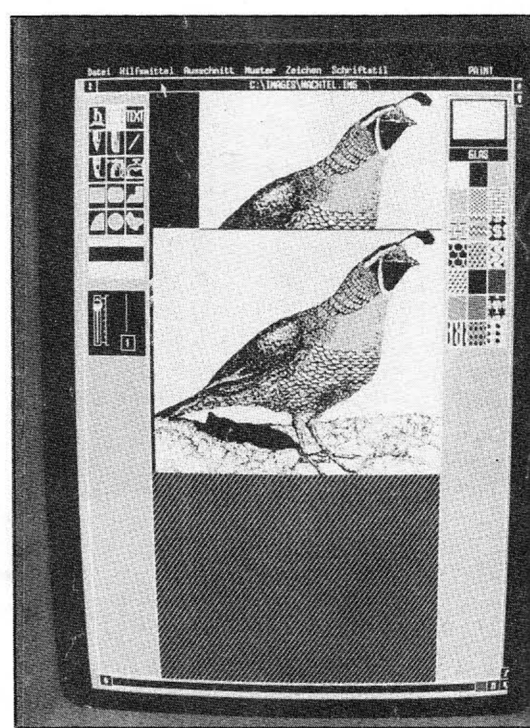
Etap-Monitore entsprechen dem neuesten Stand der Technik. Eine sehr gute Abbildungsqualität und Schärfe zeichnen diese Bildschirme aus. Störende Reflexionen sind kaum festzustellen. Eine hohe Bildwiederholfrequenz eliminiert jedes Flimmern. Die Installation dieser Bildschirme sollte auf jeden Fall einem Fachmann überlassen werden. Bei Preisen ab 5000 Mark sollte dieser Service wohl im Preis enthalten sein.

Die Palette lauffähiger, den ganzen Bilschirm nutzender Software ist umfangreich und wird ständig erweitert. Generell gesagt, ist es eine Wohltat, an einem solchen Monitor arbeiten zu können. *LS*

Bezugsquelle:
RFI Elektronik GmbH
Dohrweg 63
4050 Mönchengladbach 1
Tel. 0 21 61/6 00 60



Die Etap-Monitore bilden je nach Modell eine komplette DIN A4- oder DIN A3-Seite ab. Ein Demo-Bild aus GEM macht die Relation zu einem Standardmonitor deutlich.







Rechnen mit Binärzahlen

Inzwischen weiß es jeder, Computer rechnen nicht mit den uns gewohnten Zahlen von 0 bis 9 (Dezimalsystem), sondern mit nur zwei Zeichen oder Zuständen: der 0 und der 1. Im Grunde ist der Computer dumm, auch das ist bekannt. Er muß beim Rechnen gewissermaßen „an den Fingern abzuzählen", um zu Ergebnissen zu kommen. Finger heißt im Lateinischen *digitus*. Die zwei möglichen Zustände (1 oder 0) und die Art des Rechnens (Abzählen) verlieh der „kleinsten Einheit" für den Computer den Namen Bit (= **Bi**nary Digi**t**).

1. Stellenwerte

Für das Rechnen im Binärsystem gelten andere Regeln als im geläufigeren Dezimalsystem, dennoch gibt es Gemeinsamkeiten: Jede Ziffer bei mehrstelligen Zahlen im Dezimalsystem hat gegenüber der rechts neben ihr stehenden einen zehnfach höheren Stellenwert. Die „8" in der Zahl „48" repräsentiert 8 Einheiten (Einer). Die „4" links daneben hat den Wert von 4 Zehnern, also 40 (40 + 8 = 48).

Das Binärsystem arbeitet prinzipiell nach dem gleichen Schema, jedoch beträgt der Stellenwert jeder Binärstelle immer das zweifache des Stellenwertes der benachbarten rechten Stelle. Zur Verdeutlichung hier eine Zahlenreihe in dezimaler und binärer Form:

1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
48	110000

Hier sind schon einige Vor- und Nachteile der beiden Systeme zu erkennen: während das Dezimalsystem zur Darstellung einer Zahl weniger Stellen benötigt, braucht es andererseits viel mehr verschiedene Zeichen (zehn gegenüber zwei).

Am Beispiel läßt sich auch die Umrechnungsmethode von Binärzahlen in Dezimalzahlen nachvollziehen. Um eine Binärzahl zu schreiben, werden an die Stellen, welche addiert die Dezimalzahl ergeben, Einsen geschrieben:

32	16	8	4	2	1				
1	1	0	0	0	0	=	32 + 16	=	48
1	0	0	0	0	0			=	32
		1	0	0	0			=	8
		1	0	0	1	=	8 + 1	=	9
				1	0			=	2

Nach diesem Prinzip lassen sich beliebig große Zahlen binär darstellen. Damit sind sie vom Computer „lesbar".

Das allein genügt natürlich noch nicht, es muß auch mit diesen Zahlen gerechnet werden. Zunächst etwas Theorie hierzu; In dezimalen, beinären oder anderen Zahlensystemen können arithmetische und logische Operationen durchgeführt werden. Bei diesen Operationen treten Überträge auf, zum Beispiel bei der Addition von 3 + 7 = 10 im Dezimalsystem. Die Null wird in die Einerstelle gesetzt und die Eins in die Zehnerstelle übertragen.

Im Binärsystem wird 1 + 1 = 10 geschrieben (siehe oben). Auf diesem fundamentalen Grundsatz beruht die Leistungsfähigkeit der digitalen Datenverarbeitung.

Auch die Multiplikation schrumpft dabei auf die Tatsache zusammen, daß 1 * 1 = 1 ist. Damit ist der Rechengenauigkeit theoretisch keine Grenze gesetzt, denn es ist gleichgültig, ob 10 oder 100 mal eine Binärstelle 0 oder 1 wird; lediglich die Anzahl der verfügbaren Stllen begrenzt die Berechnung.

Subtraktionen können auf die Addition zurückgeführt werden und die Division wiederum auf die Subtraktion (dazu später mehr). Letztlich lassen sich damit alle Grundrechenarten auf die Addition zurückführen, wenn man die Bildung von Überträgen berücksichtigt, was sich mit Hilfe der Komplementbildung realisieren läßt.

2. Komplementbildung

Während die Addition auch im Rechner leicht verwirklicht werden kann, muß für die Subtraktion das Verfahren der Komplementierung angewendet werden. Unter Zuhilfenahme des Komplements einer Zahl lassen sich alle Vorgänge wieder durch Addition darstellen.

Das Komplement ist eine Zahl, die eine gegebene Zahl zu einer bestimmten Einheit ergänzt. Im Dezimalsystem wird in das Zehner- und Neunerkomplement, und im Binärsystem in das Einer- und Zweierkomplement unterschieden.

Das Zehnerkomplement bedeutet die Ergänzung jeder Ziffer einer Dezimalstelle auf 10. Die Komplementärzahl zu 1 ist 9, zu 2 ist 8 usw.

Das Neunerkomplement ist die Ergänzung jeder Ziffer auf 9. Also 1 und 8, 2 und 7 usw.

Hierzu ein Beispiel:
Von der Zahl 468 soll die Zahl 147 subtrahiert werden. Die Komplementärzahl zu 147 ist im Neunerkomplement 852.

```
1 4 7
+ + +
8 5 2
= = =
9 9 9
```

Wird das Komplement zur Zahl 468 addiert, so erhält man 1320. Abzüglich des Wertes 147 + 852 = 999 ergibt sich das Resultat mit 321.

Subtraktion:

```
  468
– 147
  321
```

Komplementäre Addition:

```
  468
+ 852
 1320
– 999
  321
```

Im Binärsystem wird mit dem Zweierkomplement gearbeitet. Dabei muß jede Binärziffer auf 10 000 (dez.16) ergänzt werden.

Doch zunächst wird einmal erklärt, wie Binärzahlen überhaupt addiert werden können.

3. Addition binärer Zahlen

Folgende Grundregeln gelten bei der Addition binärer Zahlen:

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (+Überlauf 1)
```

Die letzte Zeile erzeugt allerdings einen „Überlauf". Die 0 wird gesetzt und der Überlauf beträgt 1, der zur nächsten Stelle übertragen wird.

Hier einige Beispiele.

Dezimal	Binär
2	10
+ 2	+ 10
4	100
10	1010
+ 4	+100
14	1110
14	1110
+ 2	+ 10
16	10000

Bekanntlich arbeitet ein Computer mit Byte, welche ihrerseits aus acht Bit bestehen, die in „gepackter" Form vorliegen. Bei der natürlichen binären Darstellung laufen die Bit der einzelnen Byte ineinander über und in der Binär-Dezimal-Darstellung (das ist die Verschlüsselung von Dezimalziffern beispielsweise für den Bildschirm) wird jede Ziffer durch vier Bit, also anderthalb Byte (oder Nibble) dargestellt. Als Beispiel wird im folgenden die Zahl 121 einmal in natürlicher binärer Form dargestellt und einmal in Binär-Dezimal-Form.

Natürliche binäre Form:

1111001
= 64 + 32 + 16 + 8 + 1 = 121

Binär-Dezimal-Form:

```
0001  00010  00001
  1     2      1
```

Damit dürfte der Unterschied zwischen natürlicher binärer Form und Binär-Dezimal-Form deutlich geworden sein. Nun noch ein Beispiel einer Addition in Binär-Dezimal-Form:

Es werden die Zahlen 49 und 37 addiert (= 86).

```
   0100  1001    = dez.49
 + 0011  0111    = dez.37
   0111  1110    = Zwischenergebnis
 +          1    Übertrag
   0111  1100    = Zwischenergebnis
 +          1    Übertrag
   0111  1000    = Zwischenergebnis
 +          1    Übertrag
   0111 1 0000   Dieses Ergebnis liegt
                 außerhalb des Vier-Bit-
                 Codes, daher Subtrak-
 + 0001 –1010    tion einer dez.10 und
                 Übertrag
   1000  0110    = dez.86
```

4. Subtraktion binärer Zahlen

Die Subtraktion wird, wie schon erwähnt, in eine Addition umgewandelt. Die abzuziehende Zahl erhält dazu führende Nullen vorgestellt. Daraufhin werden alle Nullen in Einsen und umgekehrt überführt, die Zahl wird invertiert. Das Komplement einer Zahl ist also zu addieren. Die im Zwischenergebnis bei der erläuterten Addition am weitesten links stehende Eins wird gestrichen und an letzter Stelle addiert.

Beispiel:

```
10000 = dez.16
 1010 = dez.10
01010 = dez.10 mit
        führenden
        Nullen
10101 = Inversion
```

Addition:

```
  10000
+ 10101
 100101
+     1
  00110 = dez.16 –10
                 = 6
```

Und noch ein Beispiel:

68 – 23 = 45

```
1000100 = dez.68
  10111 = dez.23
0010111 = dez.23 mit
          Vornullen
1101000 = Inversion
```

Addition:

```
  1000100
+ 1101000
 10101100
+       1
  0101101 = dez.45
```

5. Division binärer Zahlen

Wie schon erwähnt, wird die Division binärer Zahlen auf eine Zwischensubtraktion und diese auf eine Addition des Komplementes zurückgeführt. Es gibt hier nur die Möglichkeit: Einmal oder keinmal ist der Divisor im Dividenten enthalten. Als Beispiel dient folgende Division:

66 : 22 = 3

```
1000010 = dez. 66
  10110 = dez. 22
0010110 = Nullen-
          auffüllung
1101001 = Inversion
```

Division:

```
  1000010 : 10110 = 11
+ 1101001
 10101011
+       1
  0101100
+ 1101001
 10010101
+       1
    10110
```

bin.11 = dez.3

6. Multiplikation binärer Zahlen

Die Gesetze der binären Multiplikation entsprechen denen der binären UND-Verknüpfung aus der Boole'schen Algebra. Hierbei gilt:

logische UND-Verknüpfung

0 UND 0 = 0
0 UND 1 = 0
1 UND 0 = 0
1 UND 1 = 1

binäre Multiplikation

0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

An dieser Stelle sind die Wahrheitstabellen der Boole'schen Algebra zu erwähnen, um die logischen UND-Verknüpfungen besser verstehen zu können. Verständlich wird diese Zuordnung aber auch dann, wenn man sich ein Kabel vorstellt, in das zwei Schalter hintereinander eingebaut sind. Bezeichnet man die Schalterstellungen mit AUS = 0 und EIN = 1, so ist ersichtlich, daß nur dann ein Strom fließen kann, wenn beide Schalter auf EIN = 1 stehen. Ist auch nur einer der beiden Schalter auf AUS = 0, so ist der Stromkreis unterbrochen, es fließt also kein Strom (=0). Hier ist schon die Verbindung vom Binärzahlensystem zur elektronischen Rechenanlage zu erkennen. Beispiel einer Multiplikation durch fortlaufende Addition:

```
      13 * 5

    1101 * 101
      1101
     0000
  + 1101
   1000001 = dez.65
```

Wie beim manuellen Rechnen mit Papier und Bleistift wird jede Produktzeile gegenüber der vorhergenenden um eine Stelle verschoben. Zu Beginn mit der Wertigkeit 1 des Multiplikators, dann 2, 4, 8 usw. von rechts nach links.

Im folgenden Schema ist die o.a. Multiplikation Schritt für Schritt unter Zuhilfenahme von vier Registern aufgezeigt. Ungefähr so läuft der Vorgang im Rechner ab. Dabei gibt es zwei Abbruchbedingungen für die Maschine: Wenn in Register A ein Überlauf auftritt, oder wenn B = 0 ist. Letzteres ist hier am Schluß der Fall. Die Multiplikation ist somit durchgeführt.

Ablauf einer Multiplikation mit Registern in einem Rechner:

Register A Multiplikant	**Register B** Multiplikator	**Register C** Test ob 1	**Register D** Ergebnis
0000001101	0000000101	1 (addiere A auf D)	0000000000 0000001101
schiebe links 0000011010	schiebe rechts 0000000010	0	
schiebe links 0000110100	schiebe rechts 0000000001	1 (addiere A auf D)	0001000001
schiebe links 0001101000	schiebe rechts 0000000000	0	

Die Register bestehen aus Speichergliedern. Jedes Speicherglied kann ein Bit darstellen. Zur Darstellung von zehn Bit sind also zehn Speicherglieder notwendig, die zu einem Register zusammengefaßt wurden. Die Register A und B sind dabei Schieberegister.

Die Stelle des Multiplikators mit der Wertigkeit 1 wird geprüft, ob sie 0

Basis^{n}	n für Basis 2	8	16	Basis^{-n}
1	0	0	0	1 0
2	1			0 5
4	2			0 25
8	3	1		0 125
16	4		1	0 062 5
32	5			0 031 25
64	6	2		0 015 625
128	7			0 007 812 5
256	8		2	0 003 906 25
512	9	3		0 001 953 125
1 024	10			0 000 976 562 5
2 048	11			0 000 488 281 25
4 096	12	4	3	0 000 244 140 625
8 192	13			0 000 122 070 312 5
16 384	14			0 000 061 035 156 25
32 768	15	5		0 000 030 517 578 125
65 536	16		4	0 000 015 258 [illegible] [illegible] 5
131 072	17			0 000 007 629 [illegible]94 [illegible] 25
262 144	18	6		0 000 003 814 [illegible] [illegible] 625
524 288	19			0 000 001 907 348 [illegible] 812 5
1 048 576	20		5	0 000 000 953 67[illegible] [illegible] 406 25
2 097 152	21	7		0 000 000 476 937 158 203 125
4 194 304	22			0 000 000 238 418 573 101 562 5
8 388 608	23			0 000 000 119 [illegible] 2[illegible]9 550 781 25
16 777 216	24	8	6	0 000 000 059 6[illegible]4 644 775 390 625
33 554 432	25			0 000 000 029 802 322 387 695 312 5
67 108 864	26			0 000 000 014 901 161 193 847 656 25
134 217 728	27	9		0 000 000 007 450 580 596 923 828 125
268 435 456	28		7	0 000 000 003 725 290 298 461 914 062 5
536 870 912	29			0 000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	10		0 000 000 000 931 322 574 615 478 515 625
2 147 483 648	31			0 000 000 000 465 661 287 307 739 257 812 5

oder 1 ist. Hier im Beispiel ist sie 1 und die Addierschaltung veranlaßt die Addition von Register A auf Register B. Bei den folgenden Schritten wird immer zuerst der Inhalt von Register A um eine Stelle nach links, der von B eine Stelle nach rechts geschoben, wie oben zu sehen ist. Wenn die rechte Position der Zahl in Register B nun 1 ist, wird die Addition von A auf D veranlaßt. Bei 0 wird sofort die nächste Verschiebung durchgeführt. Das Endergebnis steht am Schluß in Register D.

Damit wurden die vier Grundrechenarten im binären Zahlensystem erläutert. In der Regel braucht sich der Anwender nicht um diese „maschinen-nahen" Operationen zu kümmern. Es sei auch erwähnt, daß es noch andere Zahlensysteme neben Dezimal- und Binärsystemen gibt, mit denen der Computer arbeitet, beispielsweise Hexadezimal- oder Oktalzahlen.

Dipl. Ing. (FH) Oliver Rosenbaum■

* * *

Das Hexadezimalsystem

Wer sich mit Assemblerprogrammierung beschäftigen will, kommt nicht darum herum, sich auch der Hexadezimalzahlen anzunehmen. Hexadezimale Zahlen-Darstellungen erlauben, große Zahlen mit relativ wenigen Stellen darzustellen. Allerdings werden dabei mehr Zeichen benötigt.

Das Hexadezimalsystem wird auch Sedezimalsystem genannt von lat. sedecium = sechzehn. Hexadezimal kommt vom Griechischen hexa (sechs) und Lateinischen decem (zehn).

Während im Binärsystem nur zwei verschiedene Zeichen verwendet werden (0 und 1) und im Dezimalsystem bekanntlich zehn, benötigt das Hexadezimalsystem 16 Zeichen, was auch den Namen dieses Zahlen-Systems erklärt.

Schauen wir uns einmal die verschiedenen Zahlensysteme im Vergleich an:

Binär	Dezimal	Hexadezimal
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10

Das Hexadezimalsystem verwendet also die Ziffern von 0 bis 9 und zusätzlich die Buchstaben von A bis F. Es ist immer wichtig, zu wissen, in welchem Zahlensystem man sich befindet, denn beispielsweise ist (bin) 10 000 = (dez) 16 = (hex) 10.

Man gibt also immer das System mit an, und sei es nur durch einen Index (2,10 oder 16). Wir verwenden hier in Klammern gesetzte Abkürzungen.

Ein weiteres Beispiel einer Zahl in den drei Systemen:

(dez) 934 = (bin) 1110101111 = (hex) 3AF

Man rechnet die Binärzahl nach dezimal:

```
1 1 1 0 1 0 1 1 1 1
5 2 1
1 5 2   3
2 6 8 0 2 0 8 4 2 1
```

512 + 256 + 128 + 32 + 8 + 4 + 2 + 1 = 943

Binär nach hexadezimal läßt sich leicht nach der oben gezeigten Tabelle realisieren: Jede Dezimalzahl wird im Computer durch vier Bit dargestellt. Wird die Binärzahl aus dem Beispiel, von rechts beginnend, in Vierergruppen unterteilt und die entsprechenden hexadezimalen Werte aus der Tabelle abgelesen, ergibt sich:

```
11  1010  1111
3    A     F
```

also ist (bin) 1110101111 = (hex) 3AF

Genauso einfach lassen sich hexadezimale Zahlen binär darstellen. Es ist nur darauf zu achten, daß kleinere Binärzahlen mit Vornullen aufgefüllt werden.

Beispiel:
(hex) 333 = (bin)?

```
  3      3      3
0011   0011   0011
```

(hex) 333 = (bin) 1100110011

Die Umrechnung von binär nach hexadezimal (und umgekehrt) läßt sich also am einfachsten anhand von Tabellen durchführen. Dies resultiert aus der Tatsache, daß die Basis 2 des Binärsystems sich problemlos in die Basis 16 des Hexadezimalsystem umrechnen läßt.

Anders verhält es sich dagegen mit dem Dezimal- und dem Hexadezimalsystem (Basis 10 und 16).

Die Umrechnung einer Dezimalzahl in eine hexadezimale Zahl erfolgt schrittweise. Die Dezimalzahl wird immer wieder durch 16 geteilt, wobei der Rest der ersten Division die niedrigwertigste Ziffer der Hexadezimalzahl ergibt. Der Quotient wird dann wiederum durch 16 geteilt, der Rest links neben den vorhandenen Rest geschrieben, was die zweitniedrigste Stelle ergibt usw.

Beispiel: Die Zahl (dez) 246 soll hexadezimal geschrieben werden:

```
246 : 16 = 15
        Ergebnis (hex) F 6
16
086
 80
  6   ---(=hex) ---6

15 : 16 = 0
 0
15    ---(= hex)---F
```

Bei der Umwandlung von Hexadezimalzahlen in Dezimalzahlen bilden die Zeichen von 0 bis F den Multiplikanden. Der Stellenwert der Hexadezimalzahl (von links nach rechts) gibt den Exponenten an, mit dem die Basis 16 potenziert werden muß. Jedes Zeichen wird mit der entssprechenden Potenz von 16 multipliziert. Die Summe der einzelnen Produkte ergibt den Dezimalwert der Zahl.

Beispiel: Die Zahl (hex) 36F soll in eine Dezimalzahl umgerechnet werden:

$$36F = 3 * 16^2 + 6 * 16^1 + 15 * 16^0$$

36F = 3*256 + 6*16 + 15*1 = (dez) 879

Die Umrechnung einer gebrochenen Dezimalzahl in eine Hexadezimalzahl erfolgt durch Multiplizieren mit 16. Die Ziffer links des Dezimalpunkts wird als erste hexadezimale Ziffer aufgeschrieben. Die Stellen nach dem Dezimalpunkt des Produkts werden daraufhin erneut mit 16 multipliziert. Die links vom Dezimalpunkt erscheinende Ziffer ergibt die zweite Ziffer der hexadezimalen Zahl usw. Sobald ein Produkt eine ganze Zahl ergibt, ist die Umrechnung beendet.

Beispiel: (dez) 0,43 = (hex)?

```
0,43 * 16 Ergebnis (hex)    0, 6 E 1 4 7
  4 3
  5 28
  6,88 ---------------6

  0,88 * 16
     8 8
     5 28
    14,08 ---------------E

     0,08 * 16
        0 8
         48
       1,28 ---------------1

       0,28 * 16
          2 8
          1 68
          4,48 ---------------4

          0,48 * 16
             4 8
             2 88
             7,68 ---------------7
```

Mit einer Genauigkeit von hier fünf Stellen entspricht der dezimalen Zahl 0,43 die Hexadezimalzahl 0,6E147.

Günstigerweise werden der ganzzahlige Anteil einer Dezimalzahl und der gebrochene getrennt umgeformt, also zunächst die Stellen vor dem Komma, dann erst die nach dem Komma.

Bei Zusammenfassung der beiden oben gezeigten Beispiele ergäbe also

```
(dez) 246,43
(dez) 246     =   (hex) F6
(dez) 0,43    =   0,6E147

(dez) 246,43  =   F6,6E147
```

Das Hexadezimalsystem ist im Prinzip nicht schwieriger zu handhaben, als das Dezimalsystem, – wäre man letzteres nicht gewohnt und hätte man nicht lediglich zehn Finger zum „Nachrechnen". Hilfreich zum Verständnis aller hier angesprochenen Zahlensysteme ist deren gleiches Prinzip:

- es werden 2, 10 oder 16 Zeichen bestimmter Wertigkeiten verwendet;
- in allen drei Systemen zeigt die Ziffer „0", daß die Wertigkeit dieser Stelle nicht zum Gesamtwert zu addieren ist;
- die Wertigkeit einer Stelle berechnet sich immer als Produkt aus dort notierter Ziffer und der Basis, potenziert mit der Platznummer (von rechts mit 0 beginnend).

(bin) $\ldots 2^3 \; 2^2 \; 2^1 \; 2^0$

(dez) $\ldots 10^3 \; 10^2 \; 10^1 \; 10^0$

(hex) $\ldots 16^3 \; 16^2 \; 16^1 \; 16^0$

- Basis und Anzahl der verschiedenen Ziffern eines Systems sind immer gleich innerhalb eines Systems.

Umständlich ist ein Umrechnen immer, aber wenn man sich ein wenig mit dem Hexadezimalsystem beschäftigt und vielleicht auch in Assembler programmieren will, werden Zahlen wie FF oder FE vertraut. *Dipl. Ing. (FH) Oliver Rosenbaum*

PLUS-TEXT 80

Noch nicht alle Wünsche erfüllt

Der Wunsch nach einer echten 80-Zeichen-Darstellung scheint endlich in Erfüllung gegangen zu sein. Wer würde nicht gerne das auf dem Bildschirm sehen, was nachher auch so ausgedruckt wird?

PLUS-TEXT 80 wird auf Diskette für 40 Mark von BLK-Software angeboten. Statt eines Handbuchs gibt es nur ein blaues DIN-A4-Blatt, auf beiden Seiten eng beschrieben. Glücklicherweise besitzt das Programm selbst eine Menüführung, die uns alle Funktionen anzeigt.

DIE MENÜS VON PLUS-TEXT 80

1. Hauptmenü

Nach dem Laden von Diskette und einer kurzen Wartezeit, die für Initialisierungen verwendet wird, befinden wir uns im Hauptmenü. Es stehen folgende Funktionen zur Wahl:

- Eingabe
- Speichern
- Laden
- Drucken
- Zeilen einfügen
- Zeilen löschen
- Nächsten Abschnitt bearbeiten
- Einen Abschnitt zurück
- Formatieren des Textes.

Damit ist alles vorhanden, was für das Arbeiten mit einer Textverarbeitung nötig ist. Komfortabel ist eine Textverarbeitung, die zusätzliche Möglichkeiten bietet. Wir finden sie im Menü zwei, das durch Druck der Return-Taste erreicht wird.

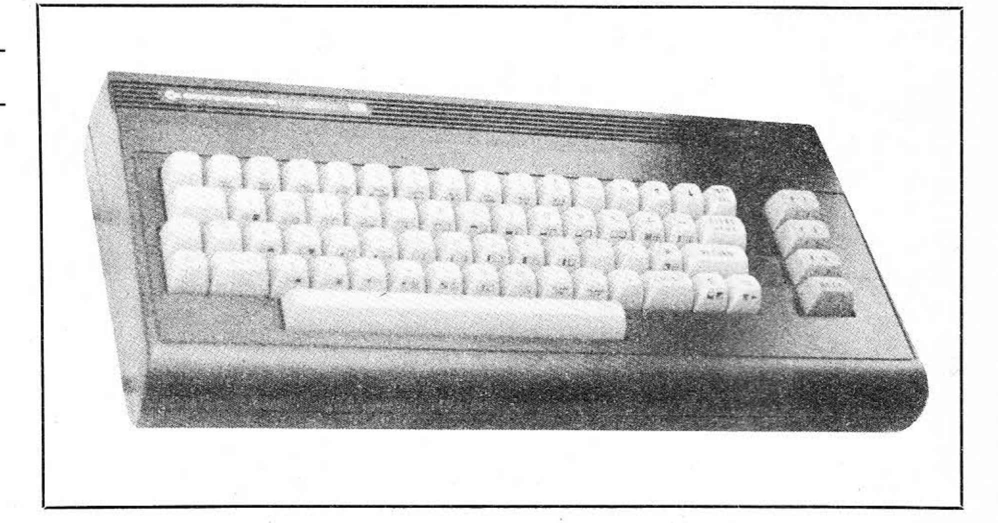
2. Menü zwei

Zusätzliche Funktionen von Menü zwei:

- Hilfsmenü drei
- Spezielle Druckeranpassung
- Text löschen
- Directory
- Datei löschen
- Zeiteingabe
- Datei umbenennen
- Diskette aufräumen
- Textbereich suchen und ersetzen
- Sortieren.

3. Hilfsmenü

Das Hilfsmenü läßt sich vom Hauptmenü und von Menü zwei aus durch Eingabe des Buchstabens h erreichen. In diesem Menü kann nichts ausgewählt werden. Es dient zur Information. Es zeigt, über welche Tasten deutsche Umlaute angesprochen werden sowie spezielle Steuertasten für Bewegungen an Zeilenanfang und Zeilenende, Textanfang und Textende. Wir erfahren, wie Tabulatoren gesetzt, gelöscht und angefahren werden können und wie spezielle Druckersteuerzeichen zu senden sind.



GLIEDERUNG NACH DISKETTEN-, DRUCKER-, TEXT- UND EDITIEROPTIONEN

Diskettenoptionen

Mit den Entsprechungen zu Directory, Scratch, Rename, Collect, Load und Save sind alle Wünsche voll abgedeckt.

Druckeroptionen

Text drucken:
Auf „Text drucken" erscheint eine Auswahl von sechs Druckern:
1. Typenrad
2. SP/180 VC
3. Star NL-10/LC-10
4. Epson
5. Matrixdrucker ohne Umlaute
6. Andere Matrixdrucker.

Der Seikosha SP/180 VC mit richtiger Dip-Schalter-Stellung ist kompatibel zum Commodore-Drucker. Wer daher einen Commodore-Drukker mit deutschem Zeichensatz besitzt, sollte Nummer zwei wählen. Nummer eins, drei und vier bringen ebenfalls Commodore-ASCII, nur werden jeweils andere Codes für die Umlaute an den Drucker gesandt. Nummer fünf ist für Commodore-kompatible Drucker ohne deutsche Umlaute gedacht. Die Umlaute werden hierbei auf dem Drucker über Einzelnadel-Ansteuerung erzeugt. Die Umlaute sehen auf den Commodore-Druckern allerdings zum Fürchten aus. Jedoch können über die spezielle Druckeranpassung die Zeichen auch selbst definiert werden, so daß ein optimales Druckergebnis erzielt werden kann. Nummer sechs bringt nicht Commodore-ASCII, sondern Standard-ASCII. Die Groß- und Kleinbuchstaben werden daher auf den Commodore-Druckern vertauscht.

Spezielle Druckeranpassung

Ein Untermenü bietet an:

- Open-Befehl ändern
- Initialisierung
- Sonderzeichen
- Druckertreiber speichern
- Druckertreiber laden
- Zeichensatz ändern
- Drucken mit Sonderzeichen und geändertem Zeichensatz.

Mit OPEN können Geräte und Se-

kundäradresse für den Drucker festgelegt werden.
Mit der Initialisierung können bis zu fünfzig Steuersequenzen für den Drucker definiert werden, wobei eine Sequenz bis zu elf Byte umfassen kann. Bei fünfzig definierten Steuer-Codes sind nach der Bedienungsangabe nur mehr durchschnittlich vier Byte pro Code möglich. Script-Plus wartet dagegen nur mit zehn selbstdefinierbaren Druckersteuerzeichen auf, wobei lediglich ein Byte definiert werden kann. Bei der Texteingabe kann dann durch einen Tastendruck, gefolgt von der Code-Nummer, veranlaßt werden, daß der Drucker beim Ausdruck die definierte Code-Sequenz empfängt. Acht Zeichen sind als Sonderzeichen definierbar. Hierbei ist gar eine Code-Sequenz von 14 Byte möglich. Wer seine deutschen Umlaute umdefinieren möchte, kann dies somit tun.
Damit nicht jedesmal eine spezielle Druckeranpassung vorgenommen werden muß, kann diese Anpassung gespeichert und wieder geladen werden.
Mit „Zeichensatz ändern" lassen sich die Codes für Klein- und Großbuchstaben verschieben. Damit kann dem Drucker zum Beispiel gesagt werden, er soll Kleinbuchstaben als Großbuchstaben und umgekehrt ausdrucken.

Besondere Textoptionen

Besondere Möglichkeiten bieten:
- Formatieren
- Suchen und ersetzen
- Sortieren.

Texte können, nachdem sie vielleicht auf die Schnelle eingegeben wurden, linksbündig, rechtsbündig oder zentriert formatiert werden. Diese Formatierung läßt sich auf gewisse Zeilen beschränken.
Buchstabenketten können gesucht und gegen andere ausgetauscht werden. Eine Liste alphabetisch zu ordnen, ist durch die Sortieroption möglich. Die Buchstabenkette, nach der sortiert wird, braucht nicht am Zeilenanfang zu stehen. Es kann die Spalte ausgesucht werden, nach der sortiert werden soll.
Ein wenig vermissen wir Blockverschieb-Operationen. Doch allzu oft werden die ja nicht gebraucht.

Das Editieren

Zeigte uns PLUS-TEXT 80 bisher noch seine guten Seiten, so können wir beim Editieren die Schwachstellen entdecken.
Die 80-Zeichen-Darstellung wird erreicht durch den HIRES-Graphic-Modus und der Verwendung einer nur vier mal acht Pixel großen Buchstabenmatrix. Da ein Pixel als Abstand zum nächsten Zeichen verlorengeht, bleiben für die Breite nur noch drei Pixel übrig. Mit einem Fernsehgerät ist hier wohl schwerlich noch etwas zu lesen. Am besten ist ein guter Monochrom-Monitor. Verglichen mit der 80-Zeichen-Karte von Kingsoft sind die Zeichen relativ gut lesbar, da auf Verschnörkelungen verzichtet und eine klare Form gefunden wurde. Dennoch ist das Lesen weitaus anstrengender als im 40-Zeichen-Modus. Für stundenlanges tagtägliches Schreiben ist diese Textverarbeitung daher nicht zu empfehlen.
Das Editieren selbst gestaltet sich nicht gerade komfortabel. Ein Zeichen einzufügen dauert vier Sekunden. So lange braucht das Programm, bis alle Zeichen um eine Stelle nach rechts verschoben sind. Nur das letzte Zeichen in der Zeile läßt sich zur Zufriedenheit löschen. Bei Zeichen, die sich mitten in der Zeile befinden, rutscht der Text rechts davon nicht nach. Beim Zwischenfügen von Text in eine Zeile, die bereits voll ist, findet am Ende kein Übertrag in die neue Zeile statt. Rückverschiebungen in die vorherige Zeile können auch nicht vorgenommen werden.
Zeilen einfügen oder löschen dauert ganze zehn Sekunden, da hierbei noch Angaben über die Anzahl der Zeilen zu machen sind und eventuell die Anwahl der Lösch- und Einfügeoption über das Hauptmenü geschieht. Bei Erreichen des unteren oder oberen Randes findet kein Scrolling statt. Es ist nur abschnittweises Blättern möglich, entweder über das Hauptmenü oder durch Direktanwahl des gewünschten Abschnitts.

FAZIT

Drucker- und Diskettenoptionen sind hervorragend verwirklicht. Das Sortieren von Listen übernimmt der Rechner durch die Sortierroutine. Die 80-Zeichen-Darstellung, obwohl sie gut realisiert wurde, ist für einen C16 oder Plus4 jedoch nicht das Wahre, besonders wenn viel damit gearbeitet werden soll. Die Editiermöglichkeiten sind ziemlich eingeschränkt und machen das Arbeiten nicht gerade komfortabel. Am besten wäre eine gute 40-Zeichen-Textverarbeitung, die es gestattet, zwischendurch den eingegebenen Text im 80-Zeichen-Modus zu begutachten. Doch so ein Programm existiert noch nicht. *a.m.* □

2500 PROGRAMMIERSPRACHEN ZUR AUSWAHL

Wie in Babylon

Der vielschichtige, schnell wachsende Markt der Programmiersprachen und ihrer Dialekte ist auch für Insider kaum noch überschaubar. Das komplexe Thema ist nicht nur für den Profi-Programmierer, sondern auch für jene geeignet, die sich einen Überblick dieser babylonischen Sprachenverwirrung verschaffen wollen. Wir beenden unsere Serie mit neunzehn weiteren Programmiersprachen — von OCCAM bis V.I.P.

49. OCCAM

Algorithmische Sprache. OCCAM entstand als Ergänzung zu dem Transputer, einem Microprozessor, der für Computer mit mehreren parallel arbeitenden CPUs (Zentraleinheiten) bestimmt ist.
Im Gegensatz zu Pascal ist die Möglichkeit, eine Aufgabe in parallellaufende Prozesse zu untergliedern, in OCCAM bereits integriert. Pascal mußte hierzu erst erweitert werden (Concurrent Pascal). OCCAM ist auch für neuere Generationen von Transputern geeignet. Während bei den Transputern der ersten Entwicklungsphasen (mit herkömmlichen Prozessoren bestückt, ein Teil der Rechenleistung der Prozessoren zur Organisation und Kommunikation zwischen diesen gebraucht wurde, kommen bei den Weiterentwicklungen spezielle Prozessoren zum Einsatz, die hierfür besser geeignet sind.
Hundert herkömmliche CPUs konnten sinngemäß niemals die hundertfache Leistung einer einzelnen erbringen, trotz der Programmiersprache OCCAM, die beide Systemkonzepte gleichermaßen unterstützt. Die volle Leistungsfähigkeit kann OCCAM bei den neuen Systemen unter Beweis stellen.
Haupteinsatzgebiet von OCCAM (und auch das der Transputer) sind technisch-wissenschaftliche Applikationen wie die Konstruktion von Filterschaltungen oder schnelle Kurvenberechnung (Fast Fourier Transformation). Auch Echtzeitgrafiken, welche naturgemäß sehr rechenintensiv sind, können hiermit realisiert werden.

50. Pascal

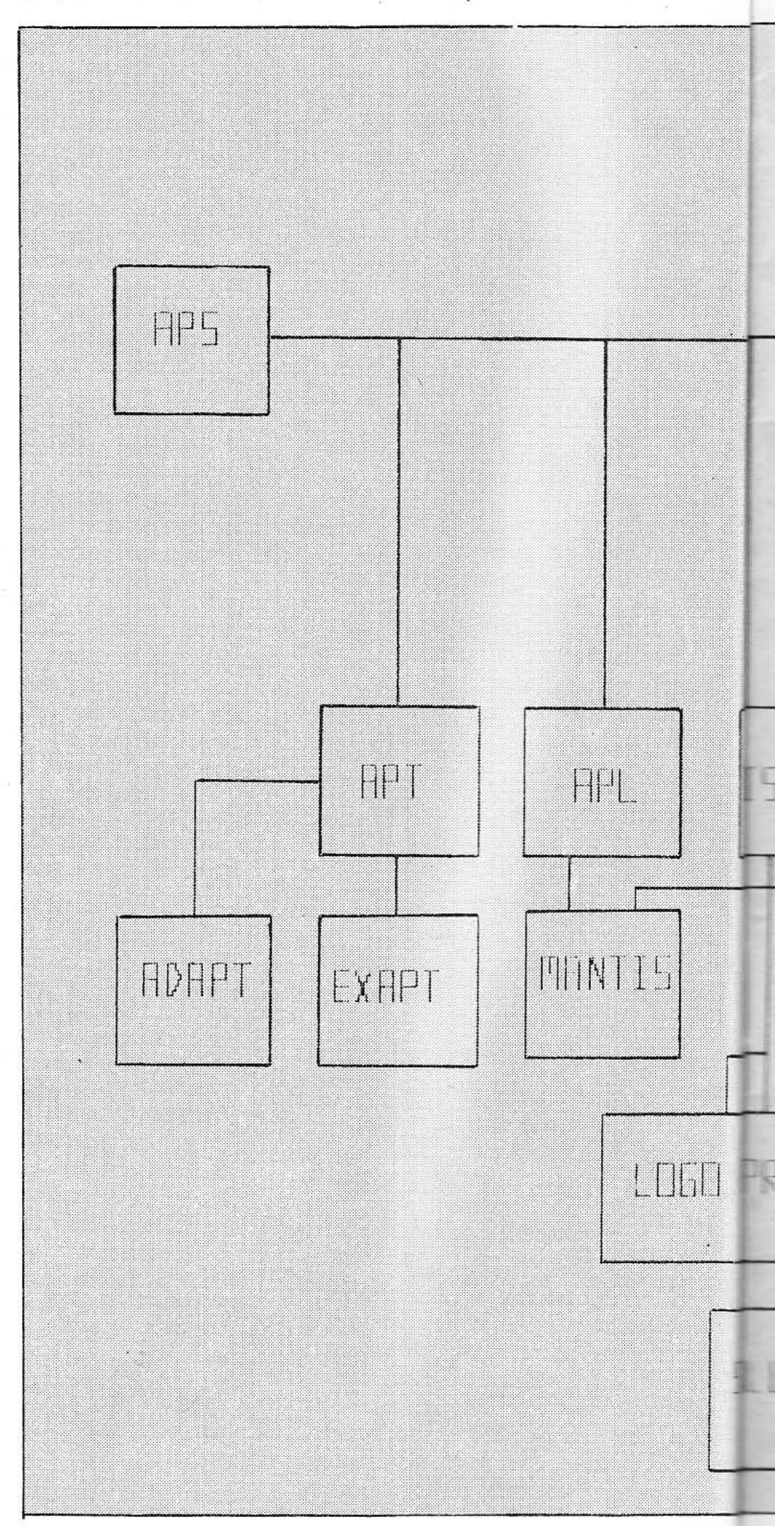
Benannt nach dem Wissenschaftler Blaise Pascal (1623 – 1663).
Pascal ist eine starke logisch strukturierte Programmiersprache, die hervorragende Hilfestellungen beim Programmieren umfangreicherer Probleme gibt.
Auf dem Markt ist Pascal seit etwa 1969.
Als Nachteil wäre zu nennen, daß Pascal für das Erstellen kleinerer Programme zu aufwendig ist.
Pascal ist eine der wenigen Sprachen, die Mengen (aus der Mengelehre) sinnvoll direkt verarbeiten kann, ohne den Programmaufwand wesentlich zu erhöhen.
Es gibt zwischenzeitlich mehrere Versionen von Pascal:

- Pascal (Jensen/Wirth-Standard),
- Pascal/Z,
- Pascal/MT+,
- UCSD-Pascal *(siehe unten)*,
- TINY Pascal,
- Concurrent Pascal (für Parallelverarbeitung),
- Pascal/M*,
- Pascal-XT (Extended Pascal),
- Pascal-SC (Pascal for Scientific Computation),
- TURBO-Pascal
- und andere.

Letztere Version dürfte wegen ihrer ausgezeichneten Verarbeitungsgeschwindigkeit die Verbreitung von Pascal wesentlich beeinflußt haben.
(Programmbeispiel siehe Tabelle 15.)

UCSD-Pascal System

Das UCSD-Pascal wurde an der University of California in San Diego entwickelt, daher auch sein



Name. Mittlerweile gibt es UCSD-Pascal für fast alle Rechnertypen, so daß die Software in Pascal sehr portabel ist.
Der Standard des UCSD-Pascal wird hier einmal anhand seines Betriebssystems dargestellt, da es sich um mehr als einen einfachen Interpreter oder Compiler handelt. UCSD-Pascal besteht aus mehreren Files (auf Diskette). Die Aufteilung in mehrere Files ist wegen des Umfangs einzelner Leistungsabschnitte notwendig. Bei Bedarf wird auf diese Files zugegriffen, daher müssen sie im Falle des Aufrufs durch den Anwender verfügbar sein; die Diskette muß sich im Laufwerk befinden. So ist beispielsweise das File mit dem Namen SYSTEM.APPLE notwendig, um das System zu booten (hier im Beispiel auf einem Apple-Computer).
Alle Systemfiles beginnen mit dem Namen SYSTEM:

SYSTEM.PASCAL
SYSTEM.CHARSET
SYSTEM.APPLE
SYSTEM.EDITOR
SYSTEM.COMPILER
SYSTEM.LIBRARY
und so weiter.

Die Namen lassen oft schon ahnen, welche Files sie bezeichnen. Dem Anwender ist es nicht möglich, Files anzulegen, die ebenfalls mit SYSTEM beginnen, damit sind die systemeigenen Files deutlich von den Anwenderfiles abgegrenzt.

Die Struktur des UCDS-Systems ist klar gegliedert in Systemeinheiten wie Editor, Compiler, Interpreter, Linker, Assembler und Filer. Die Systemkomponenten müssen also je nach Bedarf in den Rechner geladen werden. Zum Programmieren muß zunächst ein File erstellt werden, in welches mit Hilfe des Editors das Programm geschrieben werden kann. Zum Übersetzen muß dann der Compiler geladen werden.
Die Arbeit mit dem System findet also auf verschiedenen Ebenen statt. Das UCSD-System kennt drei verschiedene Filetypen zur Diskettenorganisation:
Text-Files,
Data-Filex,
Code-Files.
Der entsprechende Filetype muß bei der Namensgebung des Files mitangegeben werden, beispielsweise NAME. TEXT oder INHALT. DATA. Textfiles sind solche Dateien, die mit dem Editor beschrieben werden können (etwa Programme in Pascal). Dieses File kann abgedruckt (Programmlisting) oder compiliert werden. Der Compiler bearbeitet den Inhalt des angegebenen Textfiles und speichert das Ergebnis der Übersetzung in einem Codefile ab. Dieses enthält also den ausführbaren Code.
Datenfiles enthalten, wie ihr Name schon sagt, Daten, welche in einem Programm abgespeichert oder von diesem gelesen werden können.
Das Workfile ist das Systemfile (Text oder Code), auf das immer automatisch zugegriffen wird. Der Editor oder der Compiler öffnet – sofern nichts anderes angegeben wird – immer ein Input-File und ein Output-File. Befindet sich kein Workfile auf der Diskette und wurde auch kein anderes File angegeben, so erscheint eine Fehlermeldung.

```
Tabelle 15:

-------------------------------------------------------------
Programmbeispiel zu PASCAL
Listing
-------------------------------------------------------------
PROGRAMM SCHEIBEN (INPUT, OUTPUT);
VAR I: INTEGER
    LINKS, MITTE, RECHTS: CHAR;

PROCEDURE VERLAENGERE (I:INTEGER; LINKS, RECHTS, MITTE: CHAR);
BEGIN
   IF I .. 50 THEN
   BEGIN
      VERLAENGERE (I-1, LINKS, MITTE, RECHTS);
      WRITELN ('VON', LINKS, 'NACH', RECHTS);
      VERLAENGERE (I-1, MITTE, RECHTS, LINKS);
   END;
END;

BEGIN                                 (* HAUPTPROGRAMM *)
   I:= 3                              (* = ANZAHL DER SCHEIBEN *)
   LINKS:= 'A';
   MITTE:= 'B';
   RECHTS:= 'C';
   VERLAENGERE (I, LINKS, RECHTS, MITTE);
END.
-------------------------------------------------------------
```

```
Tabelle 16:

-------------------------------------------------------------------------
Beispiel zu PEARL Echtzeitsprache
Listing
-------------------------------------------------------------------------
AFTER 5 SEC ALL 10 SEC           Der Rechenprozeß RELAIS wird nach 5
DURING 100 MIN ACTIVATE          Sekunden, während 100 Minuten, im 7
RELAIS PRIORITZ 1                Sekundenzyklus, mit Priorität 1 belegt.
-------------------------------------------------------------------------
```

Im Directory-Beispiel sind zwei solcher Workfiles:

SYSTEM.WRK.TEXT
und
SYSTEM.WRK.CODE

Das SYSTEM.WRK. TEXT-File dient dem Editor zugleich als Ein- und Ausgabefile, denn der Editor bearbeitet nur Texte, während er Compiler Texte in Codes umformt. Daher benutzt dieser zwar das gleiche Eingabefile, braucht aber zur Ausgabe ein Code-File: SYSTEM.WRK.CODE.

Eine Diskette darf immer nur zwei Workfiles enthalten (Text und Code).
Nach dem Booten (Laden) des USCD-Systems befindet sich der Rechner in der Kommandoebene, von der aus Direktbefehle gegeben und Dienstprogramme gestartet werden können. Der Rechner erwartet einen Befehl:

COMMAND: Edit, Run, File, Comp, Link, Xecute, Assem, Debug.

Weitere Befehlsmöglichkeiten erfährt man beim Drücken auf die Taste „?“. Es erscheint:

COMMAND: User restart, Initialize, Halt, Swap, Make exec.

Diese Befehle sind sogenannte Direktbefehle, also Teil des Betriebssystems, während die Befehle der oberen Zeile Aufrufe für Dienstprogramme sind. Diese müssen erst von der Diskette nachgeladen werden.
Anders als in BASIC bewirkt Run hier nicht lediglich einen Programmstart. Run ist selbst ein Dienstprogramm, welches den Compiler aufruft und startet. Dieser compiliert daraufhin das Workfile. Der Linker wird bei Bedarf automatisch dazugeladen. Das so übersetzte und gebundene Programm wird anschließend noch gestartet.
Run wird also nur für noch nicht compilierte Programme benötigt. Der Befehl, welcher dem BASIC-Run entspricht, wäre hier der „eXecute“-Befehl.
Run erleichtert das Austesten von Programmen, da bei der Fehlersuche nicht jedesmal der Compiler von Hand aufgerufen werden muß.
Neben dieser Möglichkeit kann der Pascal-Compiler auch direkt aufgerufen werden.
Der Compier übersetzt den Inhalt des Workfiles (Text) in direkt ausführbaren Code und speichert diesen in dem Workfile SYSTEM.WRK.CODE ab. Ist eines der beiden Work-

files auf der Diskette nicht vorhanden, oder sollen andere Files benutzt werden, so kann dies optional angegeben werden. Erst compilierte Programme können mit „eXecute" gestartet werden.
Der Pascal-Linker ermöglicht die Einbindung von Assembler-Routinen in ein Pascal-Programm. Diese Routine muß im Pascalprogramm als „external" deklariert sein. External veranlaßt das System, vor Ausführung des Programmes die einzubindende Routine aufzusuchen und zu linken. Der Compiler meldet dies am Bildschirm. Beim Run-Befehl wird der Linker automatisch aufgerufen.
UCSD-Pascal erlaubt auch eine Art Batch-Betrieb: Ein Exec-File, welches mit „M" erzeugt wird, kann Systembefehle enthalten, die bei Start des Exec-Files ausgeführt werden. Dabei ist es gleichgültig, ob es sich um Direktbefehle oder Dienstprogramme handelt. Wie bei manchen anderen Betriebssystemen kann somit ein Batchfile erstellt werden, ein File, das Stapelverarbeitung zuläßt (Batch = Stapel).
Der Name rührt noch aus der Lochkartenzeit. Man hatte die Möglichkeit, Lochkarten mit verschiedenen Direktbefehlen aufeinanderzustapeln und diese gebündelt dem Computer einzugeben. Die Bearbeitung der Befehle erfolgt in der vorgegebenen Reihenfolge. Sprünge oder Verzweigungen sind nicht möglich.
Das UCSD-Pascal-System hat in seiner Struktur große Ähnlichkeit zu verbreiteten Betriebssystemen größerer Rechner. Ein markanterer Vorteil ist aber die Standardisierung für verschiedene Rechner und die Verbindung von Sprache und System zu einer Einheit. Erst dadurch ist es möglich, echte portable Software zu entwickeln. Zwar gibt es inzwischen viele Compiler für Pascal, aber das UCSD-System konnte sich behaupten.

51. Pearl
Process and **E**xperiment **A**utomation **R**ealtime **L**anguage.
Wie der Name dieser Programmiersprache schon sagt, ermöglicht sie das Realtime Processing, also die Echtzeit-Verarbeitung: Das Programm läuft in der gleichen Zeit wie ein Prozeß. Der Rechner arbeitet also parallel zu einem technischen Ablauf. Die Daten, die aus diesem technischen Ablauf (Prozeß) gewonnen

ECHTZEIT-VERARBEITUNG

werden, werden simultan verarbeitet. Aufgrund dieser Daten kann der Rechner in den Prozeß in Form einer Korrektur des Prozeß-Ablaufs eingreifen, oder für die Fortführung des Prozesses wichtige Entscheidungen treffen (Prozeßdatenverarbeitung).
Bei der Konzeption von Pearl wurde besonderen Wert darauf gelegt, daß Echtzeit-Sprachelemente in den Sprachkern von Pearl aufgenommen wurden – anders als bei den meisten übrigen höheren Programmiersprachen –, die über solche spezifischen expliziten Sprachmittel nicht verfügen. Meist werden dort beispielsweise in Unterprogrammaufrufen Funktionen des Betriebssystems organisiert.
(Programmbeispiel siehe Tabelle 16.)
Die Zeichenketten-Verarbeitung in Pearl ist unzureichend, obwohl diese Sprache fast ausschließlich auf digitalen Großrechnern zum Einsatz kommt.
Hingegen sind die Strukturen für die Ein- und Ausgabe, sowie die Sprachmittel für Algorithmik ähnlich denen anderer höherer Programmiersprachen. Einflüsse von Algol 68 und Pascal sind augenfällig. Die Syntax ist weitgehendst an PL/1 angepaßt.
Versuche der Implementierung von Pearl auf kleineren Anlagen scheiterten lange an dem für damalige Verhältnisse großen Speicherplatzbedarf.
Inzwischen stehen auch bei kleinen Computern Speicherkapazitäten zur Verfügung, die eine Installation von Pearl zuließen. Jedoch nur für wenige Kleinrechner oder Microcomputer gibt es Implementierungen (zum Beispiel RTOS-Pearl).
Eines der Hauptmerkmale dieser „Prozeßrechnersprache" ist die Trennung in resistente und nichtresistente Prozeduren.
Pearl wurde gemeinsam von deutschen Wissenschaftlern, dem VDI/VDE, industriellen Anwendern, deutschen Prozeßrechnerherstellern und Softwareproduzenten entwickelt und wurde erstmals 1980 in der DIN 66.253, Teil 2: „Informationsverarbeitung; Programmiersprache Pearl, Full Pearl", festgelegt, allerdings in englischer Sprache mit der Absicht, den internationalen Anwenderkreis zu berücksichtigen.
(Programmbeispiel siehe Tabelle 17.)
Ähnlich anderer höherer Programmiersprachen (etwa Cobol) wird ein Pearl-Programm in verschiedene Divisions unterteilt. Diese Einheiten sind unabhängig voneinander compilierbar. Die getrennt übersetzten Programmteile (Module) werden mit sogenannten Global-Größen miteinander verbunden.
Ein fertiges Modul besteht im allgemeinen aus zwei Divisions: der System-Division, welche die Systemumgebung beschreibt und der Problem-Division, der Problemlösung einer bestimmten Aufgabe.
Im System-Teil eines Pearl-Programms werden die peripheren Geräte und deren Verbindungen beschrieben; insbesondere werden Eingangs- und Ausgangskanäle definiert. Es kann hier weiterhin festgelegt werden, ob die Übertragung auf diesen Kanälen (also von oder zu den peripheren Geräten) digital oder analog erfolgen soll. An dieser Stelle werden die Vorteile der Programmaufteilung in Problem- und System-Division am deutlichsten: Sollen bei bestehendem System einzelne Teile der Peripherie gegen verbesserte Gerätetypen ausgetauscht oder ergänzt werden, müssen lediglich einzelne Teile der System-Division geändert werden. Nämlich nur die dieses Gerät beschreibenden Programmzeilen, nicht jedoch alle Befehle im Programm, welche das entsprechende Gerät ansprechen, wie es in manchen anderen Programmiersprachen der Fall sein kann.

SYSTEM- UND PROBLEM-DIVISION

Bei der Übertragung des Pearl-Programms auf andere Rechnertypen oder Anlagenkonfigurationen kann die System-Division komplett ausgetauscht werden, die Problem-Division kann dabei unverändert bleiben.
Hierdurch ist der System-Teil von Pearl durchaus vergleichbar mit einer Job-Control-Language üblicher Compiler.
In der Problem-Division wird das eigentliche Problem formuliert, das mit Hilfe des Programms bewältigt werden soll. Der Problem-Teil besteht aus verschiedenen Tasks, die eine ähnliche Struktur haben können wie Haupt- und Unterprogramme anderer höherer Programmiersprachen. Diese Tasks aktivieren sich gegenseitig in vorgegebener Abfolge oder gleichzeitig, je nach Anforderung. Die erste Task eines Pearl-Programms wird von außen aktiviert (Programmstart).

Pearl erlaubt die Verwendung von sechs verschiedenen Datentypen:

FIXED
ganze Zahlen,
FLOAT
Gleitkommazahlen,
CHARACTER
Zeichenketten,
BIT
Bitmuster (Kette),
CLOCK
Zeitpunkte,
DURATION
Zeitdauer.

Im Programm verwendete Variablen und Konstanten müssen als einer der oben aufgeführten Datentypen vereinbart werden. Ähnlich wie in Pascal lassen sich in Pearl Felder und Zuordnungen von Variablen bilden. Felder können wieder Felder enthalten und getrennt angesprochen werden. Hierdurch ist es in Pearl möglich, die Daten hierarchisch zu strukturieren, Gruppen innerhalb von Feldern verschiedenen Prioritätsstufen zuzuordnen.
Pearl-Prozeduren steuern die Übergabe von Parametern und Konstanten. Die Übergabe eines Parameters kann etwa von dessen Wert oder seiner Adresse (Identität) abhängig gemacht werden. Man unterscheidet:

- Initial-Prozedur: Der aktuelle Parameter wird einfach kopiert;
- Identical-Prozedur: Im Gültigkeitsbereich des Prozedurblocks (Modul oder Task) wird ein neuer Name für den Parameter vereinbart;
- Returns sind Funktionsprozeduren, die eine Größe zurückliefern;
- Inline bewirkt eine Direkteinfügung des entsprechenden Prozedur-Parameters.

In Pearl sind Sondervereinbarungen möglich. Dem Benutzer wird gestattet, sowohl neue, abstrakte Datentypen (neben den oben genannten) als auch neue Operationen zu definieren. Besonders interessant und effektiv ist diese Möglichkeit der Sonderfunktionsgestaltung im Zusammenhang mit den oben genannten Feldern.
Die E/A-Struktur von Pearl besteht aus einem Netzwerk von „Datenstationen“ und „Kanalumsetzern“.

```
Tabelle 17:

----------------------------------------------------------------------
Programmbeispiel zu PEARL
Listing
----------------------------------------------------------------------
MODULE;                                  /* EXAMPLE PRESSURE CONTROL */

      SYSTEM;
            MULTIPLEXOR                  CPU*5;
            DIGITALOUT(0)                MULTIPLEXOR*0;
            ANALOGIN(0)                  MULTIPLEXOR*9;
            VALVE         :              DIGITALOUT(0)*0,2;
            LAMP          :              DIGITALOUT(0)*3,1;
            DISCHARGE     :              DIGITALOUT(0)*6,1;
            PRESSURESENSOR1:             ANALOGIN(0)*1;
            PRESSURESENSOR2:             ANALOGIN(0)*3;
            READY:                       INTERRUPTINPUT*0;
            ALARM:                       INTERRUPTINPUT*2;

      PROBLEM;
                           SPECIFY VALVE DATION OUT BASIC DIM() TFU,
                                   (LAMP, DISCHARGE) DATION OUT BASIC,
                                   (PRESSURESENSOR*1, PRESSURESENSOR*2)
                                   DATION IN BASIC DIM() TFU,
                           DECLARE (PRESS1, PRESS2, MEANPRESS) FLOAT,
                                   OPENVALVE BIT(2)  INITIAL('10'B),
                                   CLOSEVALVE BIT(2) INITIAL('10'B),
                                   (TURNON, TURNOFF)
                                   BIT INITIAL ('1'B, '='B);

            START:         TASK GLOBAL;     /* ACTIVATED FROM OUTSIDE */

                           EVERY 1 SEC ACTIVATE PRESSURECONTROL;
                           WHEN ALARM ACTIVATE ALARMING;
                           END;                      /* OF TASK STRAT */
            PRESSURECONTROL: TASK PRIORITY 5;
                           TAKE PRESS 1 FROM PRESURESENSOR1;
                           TAKE PRESS 2 FROM PRESURESENSOR2ä
                           MEANPRESS = (PRESS1 + PRESS2)/2;
                           IF MEANPRESS = .30
                                THEN SEND OPENVALVE TO VALVE;
                                ELSE IF MEANPRESS = .20 THEN SEND
                                CLOSEVALVE TO VALVE;
                                   FIN;
                           FIN;
                           END;           /* OF TASK PRESURECONTROL */

            ALARMING:      TASK PRIORITY 3;
                           SEND TURNON TO LAMP;
                           SEND TURNON TO DISCHARGE;
                           WHILE MEANPRESS = .20
                                   REPEAT
                                   AFTER 2 MIN RESUME;
                           END;                               /* LOOP */
                           SEND TURNOFF TO LAMP;
                           SEND TURNOFF TO DISCHARGE;
                           END;                   /* OF TASK ALARMING */
----------------------------------------------------------------------
```

Datenstationen verkörpern allgemein alle Peripheriegeräte oder deren Kanäle. Jede Datenstation besteht aus maximal vier Kanälen: dem Datenkanal, dem Steuerkanal, dem Unterbrecherkanal und dem Signalkanal.
Die einzelnen Kanäle müssen im Programm spezifiziert werden.

Kanalumsetzer schaffen die Möglichkeit der Anpassung verschiedenartiger Peripheriegeräte auf einheitliche Formate zur Kommunikation mit der Zentraleinheit oder untereinander. Kanalumsetzer arbeiten wie Interfaces. Sie setzen die Daten innerhalb eines Kanals in einer prozedurartigen Routine um.
Zur Pearl-Syntax gehört eine Reihe von Sprachelementen, die es ermöglichen, beispielsweise parallel ablaufende Prozesse zu steuern und sowohl termin- und zeitgerechte Abläufe zu aktivieren.
Auf die vielschichtigen Befehlsstrukturen kann hier nicht eingegangen werden, aber im obigen Beispiel wird ziemlich deutlich, welche Realtime-Möglichkeiten Pearl besitzt.
Pearl ist für Anwender geschaffen worden. Gegenüber Systemimplementierungssprachen wie Concurrent Pascal, Modula oder Ada ist Pearl leichter erlernbar und bietet zudem große Vorteile.
Die Sprachelemente zur Formulierung algorithmischer Zusammenhänge und Abläufe entsprechen dem Standard moderner Programmiersprachen (so etwa Pascal, PL/1).
Pearl hat durch die Aufteilung in Divisions ein hohes Maß an Portabilität und Dokumentationswert. Strukturiertes Programmieren wird unterstützt. Pearl ist in vielen Anwendungsbereichen eine echte Alternative zur Assembler-Programmierung.

52. PILOT

Programmed Inquiry Learning Or Teaching.
PILOT ist eine Dialogsprache (siehe auch BASIC) und wurde von Prof. John Starkweather entwickelt. PILOT hat nie den Bekanntheitsgrad von BASIC erreichen können. Gründe hierfür sind unter BASIC (im ersten Teil unserer Serie) nachzulesen.

Im Gegensatz zu BASIC ist PILOT aber eine Dialogsprache ohne Kompromisse. Sie wird oft als *die* Dialogsprache schlechthin bezeichnet. Heutigen Versionen, wie dem Utah-PILOT (auch für IBM-PC kompatible Rechner unter MS-DOS erhältlich), liegt der PILOT-73-Standard zugrunde, – ständig erweitert und verbessert.
PILOT ist für Anfänger geschrieben und leicht erlernbar und kann daher auch ideal für Lernzwekke eingesetzt werden.
Als Dialogsprache ist sie im besonderen Maße für interaktive Anwendungen geeignet. Hierzu werden großartige Hilfestellungen in der Syntax geboten: spezielle Dateneingänge, programmierte Instruktionen, Testroutinen und anderes.
Interaktive Anwendungen liegen dann vor, wenn im besonderen Maße Wert auf den Dialog zwischen Mensch und Maschine gelegt wird (was oft problembedingt sein kann).
PILOT hat ein recht einfaches Format und eine leicht verständliche Syntax.

53. PLANKALKÜL

Eine der ersten universellen algorithmischen Sprachen überhaupt. Entwikkelt wurde sie schon 1945 von Dr. Ing. e.h. Konrad Zuse, dem Erbauer der ersten programmgesteuerten Rechenanlage.
PLANKALKÜL war wegweisend für die Entwicklung aller symbolischen Programmiersprachen.

54. PL/1

Programming Language Nr. 1
PL/1 könnte man als Synthese aus Fortran, Algol und Cobol bezeichnen, was ihren Anwendungsbereich sowohl in den technisch-mathematischen als auch in den kaufmännischen Bereich legt (Mischsprache). Hier wurde versucht, die aus der Erfahrung mit diesen Sprachen erkannten Mängel zu vermeiden und die Vorteile in einer gemeinsamen Sprache zu vereinen.

```
Tabelle 18:
---------------------------------------------------------------------------
Programmbeispiel zu PL/I:
Listing
---------------------------------------------------------------------------
      BLOCK1: PROCEDURE OPTIONS(MAIN);
              DECLARE A DECIMAL FIXED(5,2),
                      B BINARY FLOAT(24),
                      C CHARALTER(20), VARYING;
              A = 2.34;
              B = 4.56E+4;
              C = 'BEISPIEL';
              PUT SKIP(2) LIST('BLOCK 1, ZEILE 9:'):
              PUT SKIP LIST('A =' , A,' B =' ,B,' C = ' ,C);

         BLOCK2: BEGIN;
                 DECLARE A BINERY FLOAT(18) EXTERNAL,
                         X BINERY FIXED(15),
                 A = 2.875E+3;
                 B = 5.625E-3;
                 X = 275;
                 PUT SKIP(2) LIST('BLOCK 2, ZEILE 18:'):
                 PUT SKIP LIST('A =' ,A,' B =' ,B,' C =' ,C,' X =' ,X,);
                 END BLOCK2;

              PUT SKIP(2) LIST('BLOCK 1, ZEILE 22:'):
              PUT SKIP LIST('A =' ,A,' B =' ,B,' C =' ,C);

         BLOCK 3: BEGIN;
                  DECLARE A BINARY FLOAT(!() EXTERNAL,
                          C CHRACTER(!=) VARYING,
                          X BINARY FIXED(!%);
                  X = 386;
                  C = ' BEISPIEL 1';
                  PUT SKIP(2) LIST('BLOCK 3, ZEILE 31:'):
                  PUT SKIP LIST('A =' ,A,' B =' ,B,' C =' ,C' X =' ,X,);
                  END BLOCK3;

              PUT SKIP(2) LIST('BLOCK 1, ZEILE 35:'):
              PUT SKIP LIST('A =' ,A,' B =' ,B,' C =' ,C);

              END BLOCK1;
```

Von Algol wurde die strenge Einteilung eines Blocks in Vereinbarungs- und Anweisungsteil, die Blockstruktur sowie die formatfreie Ein- und Ausgabe, die Ablaufstrukturen, Records, Pointer und das Prozedure-Konzept mit rekursivem Aufruf übernommen. Cobol lieferte die Dezimalarithmetik, Picture-Formatierung, Textverarbeitung, Datenstrukturierung und eine umfangreiche Dateiverwaltung. Aus Fortran kommen die impliziten Laufanweisungen, numerische Standardfunktionen, die Functions-Subroutine und das Prinzip der Common-Bereiche (siehe hierzu auch unter Fortran).
Der Nachteil: PL/1 ist in vollem Umfang nur auf größeren Rechenanlagen anwendbar, da der benötigte Speicherplatz wesentlich größer ist als der bei anderen Programmiersprachen.
Auch die Syntax ist umfangreicher, was das Erlernen von PL/1 erschwert.
Vorteile: PL/1 ist flexibler in den Anwendungen und problemloser bei der Verarbeitung großer Datenmengen innerhalb der Rechenanlage. PL/1 vereint die Vorteile verschiedener anderer Programmiersprachen in sich: auf der einen Seite gute Texthandhabung und Dialogfähigkeit, auf der anderen Seite die mathematischen Möglichkeiten der genannten Programmiersprachen.
Der modulare Aufbau von PL/1 gestattet es dem Anfänger, zunächst einen kleinen Teil der Syntax zu erlernen und sich dann schrittweise den gesamten Sprachumfang anzueignen.
Vieles wird vom PL/1-Compiler übernommen, wie zum Beispiel die Speicherplatzorganisation, Datenumwandlungen, Formatierung der Ein- und Ausgabe und anderes.
Man spricht hierbei vom Default-Konzept. Der PL/1-Sprachumfang ist so umfassend, daß fast alle Anwendungsgebiete abgedeckt werden können. So muß nicht bei wechselnden Anforderungen auch die Programmiersprache gewechselt werden.
PL/1 unterstützt Mehrbenutzer-Betriebssysteme durch mehrstufige Passwortabfrage und Dateischutz bis zum Sperren.
Integriert sind außerdem

FLEXIBEL UND UMFASSEND

umfangreiche Möglichkeiten zur Behandlung von Ausnahme- und Fehlersituationen sowie Overlaytechniken.
PL/1 wurde zum ersten Mal 1965 vorgestellt. Seitdem wurde sie, wie viele andere Programmiersprachen auch, weiterentwickelt. Da dies aber nicht durch einen neutralen Ausschuß geschah (ANSI), gibt es heute mehrere PL/1-Dialekte. Erst 1979 wurde PL/1 in einer Norm vereinheitlicht:

- ISO 6160: „Programming Language PL/1" (international);
- DIN 66.255: „Informationsverarbeitung; Programmiersprache PL/1", Ausgabe 1980.

Jedoch in englischer Sprache, da man eine Abweichung bei der Übersetzung ins Deutsche befürchtete, was unter allen Umständen vermieden werden sollte, denn die Deutsche Industrie-Norm sollte fachlich der internationalen (ISO) Norm entsprechen.
PL/1 gehört heute zu den am meisten verbreiteten Programmiersprachen und ist auch für die Simultanverarbeitung geeignet.
Simultanverarbeitung liegt

dann vor, wenn ein Programm gleichzeitig mit mehreren peripheren Geräten (Drucker, Dateien...) arbeitet. Hierbei entsteht weniger Wartezeit, da die Peripherie einer Anlage sehr viel langsamer ist als die Zentraleinheit.
Steuereinheiten in den Kanälen ermöglichen den gleichzeitigen Einsatz mehrerer Geräte.
Ein weiterer Vorteil von PL/1 liegt in den vorhandenen Sprachbestandteilen zur Gliederung von Programmen.
In PL/1 werden Unterprogramme als Prozeduren geschrieben, die wiederum von anderen Prozeduren aufgerufen werden können. Komplizierte Aufgabenstellungen können so in logisch unterteilte Abschnitte (Module)

MODULTECHNIK

untergliedert werden. Dies verbesserte die Effezienz eines Programms erheblich, vor allem in Hinblick auf die Transparenz des Lösungswegs. Korrekturen und Änderungen (die Programmpflege) werden dadurch erheblich erleichtert.
Alle Prozeduren, die nach dem Start des Haupt-Programms bis zur Rückgabe der Ablaufsteuerung an das Betriebssystem aktiviert werden, bilden das Gesamtprogramm.
Dieses kann aus internen und externen Prozeduren bestehen. Das Hauptprogramm ist dabei immer eine externe Prozedur, die durch den Zusatz Options(Main) gekennzeichnet wird, alle anderen Prozeduren sind Unterprogramme.
Interne Prozeduren sind Bestandteile anderer Prozeduren (übergeordnete Prozedur) und werden mit dieser gemeinsam übersetzt und gebunden. Sie können nur mit der übergeordneten Prozedur verwendet werden.
Im Gegensatz hierzu werden externe Prozeduren separat übersetzt und bilden eigenständige Module. Diese können in Bibliotheken abgelegt und bei Bedarf an Hauptprogramme angebunden werden.
PL/1 gibt es in der Zwischenzeit auch für viele Klein- und Micro-Computersysteme unter CP/M oder MS-DOS.
Bekannt wurde noch PL/9900, eine Abwandlung von PL/1, entwickelt von Texas Instruments für deren Rechnerkonzept 9900. Inzwischen hat aber fast jeder Hardware-Hersteller eigene Implementationen von PL/1 für seine Maschinen im Angebot, sofern die Rechner für PL/1 geeignet sind.
Eine interessante Neuentwicklung stellte auch PL/M dar, eine spezielle, abgemagerte Version für Micro-Computer.
(Programmbeispiel siehe Tabelle 18.)

55. Prolog

PROgramming in **LOG**ic.
Prolog ist eine deklarative Programmiersprache, wie sie zur Lösung von Problemen aus dem Bereich der künstlichen Intelligenz (KI) eingesetzt werden.
Deklarative Sprachen beschreiben die vom Programm zu lösende Aufgabe, der Lösungsweg (Algorithmus) muß nicht definiert werden. Daher spricht man auch von logischer oder symbolischer Programmierung.
Die in Prolog bevorzugt verwendeten Datenstrukturen sind einfache und verkettete Bäume. Der Begriff des Datentyp im Sinne konventioneller Programmiersprachen existiert in der logischen Programmierung nicht.
Verschiedene Prolog-Compiler verlangen jedoch eine Deklarierung von Variablen für die Compilierung des Quellcodes.
Statt Datentypen werden in Prolog Domain-Typen verwendet:

integer	**ganze Zahlen**
real	**reelle Zahlen**
char	**Zeichen**
string	**Zeichenkette**
symbol	**Symbol**
file	**Datei-Name.**

Integerzahlen liegen zwischen −32.768 und +32.767.
Reelle Zahlen sind durch einen Dezimalpunkt gekennzeichnet, auch wenn keine Nachkommastellen auftreten.
Größere Zahlen werden in exponentieller Schreibweise dargestellt. Die Char-Variablen können einzelne Zeichen enthalten. Die Zeichen werden mit Hochkommas geschrieben. Zeichenketten (Strings) sind beliebige Zeichenkombinationen, auch sie werden durch Hochkommas abgegrenzt.
Symbol-Variablen beginnen mit Kleinbuchstaben oder sind Zeichenfolgen, die von Hochkommas eingeschlossen sind. Symbol und String werden in Prolog unterschiedlich verarbeitet.
File-Variablen ermöglichen die Definition symbolischer Dateinamen, welche als Argumente in Prädikaten verwendet werden können.
Jedes Programm darf aber nur einen Domaintyp enthalten, der zur Gruppe „File" gehört.
In der Deklarierung können mehrere Objekte des gleichen Domain-Typs zusammengefaßt werden. In Prolog ist es möglich, für die Definition einer Domain diese selbst zu verwenden. Dadurch entsteht ein rekursives Objekt.
Standard-Domaintypem müssen nicht deklariert werden.
Ein Prolog-Programm ist als Sammlung von Fakten und Regeln auch eine Datenbank. Durch Einfügungen und Löschungen während der Ausführung ist sie dynamisch.
Standard-Prolog erlaubt den Austausch von Fakten und Regeln. Manche Prolog-Dialekte wie Turbo-Prolog lassen nur den Austausch von Fakten zu.
Die Anordnung von Fakten und Regeln, sowie der Datensätze ist in Prolog sehr wichtig für die Ausführung des Programms.
Je nach Anordnung kann ein Programm auf verschiedene Weisen interpretiert werden und es kann zu völlig unterschiedlichen Ergebnissen kommen.
An dieser Stelle sollte vielleicht einmal etwas Grundsätzliches zu den sogenannten KI-Sprachen gesagt werden.
Wie schon erwähnt, handelt es sich hier um deklarative Programmiersprachen. Im Gegensatz zu den herkömmlichen Sprachen wird nur ein Problem, nicht aber der komplette Lösungsweg beschrieben.
In algorithmische Sprachen wie Fortran, Cobol, BASIC, Pascal und andere müssen Lösungswege für alle möglichen Fälle bei der Programmierung definiert werden, was sehr aufwendig sein kann und damit auch als Qualitätsmerkmal eines Programms gilt.
Logische oder deklarative Programmiersprachen erlauben aber die Festlegung von Fakten (Facts) und Regeln (Rules), die für die eigentliche Lösung des Problems notwendig sind.
Also nicht der Programmie-

KÜNSTLICHE INTELLIGENZ

rer löst das gestellte Problem, sondern das Programm. Die Hauptarbeit des Programmierers besteht darin, die „Facts" und „Rules" korrekt und vollständig zu formulieren.
Bei Prolog ist das oben Gesagte leider nur zum Teil verwirklicht worden. Auch Prolog kann nicht ganz ohne explizite Anweisungen (Algorithmen) auskommen. Bei den konventionellen Programmiersprachen ist dieser Algorithmus das Programm selbst. Bei Prolog ist der benötigte Algorithmus inhärenter Bestandteil des Interpreters oder des Compilers. Aus diesem Grunde wurde schon mehrfach versucht, Prolog den deklarativen Charakter abzuerkennen. Sicher ist Prolog keine reine deklarative Programmiersprache, sondern eher eine Mischform,

wie viele höhere Programmiersprachen.

Um einen Eindruck zu gewinnen, wie ein Prolog-Programm aussieht, in Tabelle 19 ein Ausschnitt, geschrieben in Turbo-Prolog.

Es gibt viele Prolog-Implementierungen, von denen sich die meisten an den De-Facto-Standard halten.

Prolog-Programme werden mit einem gewöhnlichen Editor erstellt. Manche Implementationen erlauben die Syntax-Überprüfung schon während des Editierens.

Zudem werden sogenannte Software-Interfaces angeboten, die Schnittstellen zu anderen Programmiersprachen darstellen (etwa zu C).

Grundsätzlich zu unterscheiden ist zwischen Prolog-Interpreter und Compilern. Vor- und Nachteile dieser Übersetzungsmethoden wurden schon hinreichend besprochen. Sie gelten für alle in Frage kommenden Programmiersprachen.

```
Tabelle 19:

--------------------------------------------------------------------------------
Programmbeispiel zu Turbo-PROLOG
Listing
--------------------------------------------------------------------------------
domains
       position = integer*
       nummer = integer
       zugliste = nummer*

Predicates
       regel ( nummer, position, position)
       wende (integer, integer)
       endposition (position)

       postcheck (position)
       gültig (podition)

       löse (position, integer, zugliste)
       run

clauses
       regel(1,(x1,x2,x3,x4,x5,x6,x7,x8,x9),(y1,y2,y3,y4,y5,y6,y7,y8,y9,)
       if wende(x1,y1), wende(x2,y2), wende(x4,y4), wende(x5,y5),
       x3 = y3, x6 = y6, x7 = y7, x8 = y8, x9 = y9
       regel(2,(x1,x2,x3,x4,x5,x6,x7,x8,x9),(y1,y2,y3,y4,y5,y6,y7,y8,y9,)
       if wende(x1,y1), wende(x2,y2), wende(x3,y3),
       x4 = y4, x5 = y5, x6 = y6, x7 = y7, x8 = y8, x9 = y9
       regel(3,(x1,x2,x3,x4,x5,x6,x7,x8,x9),(y1,y2,y3,y4,y5,y6,y7,y8,y9,)
       if wende(x2,y2), wende(x3,y3), wende(x5,y5), wende(x6,y6),
       x1 = y1, x4 = y4, x7 = y7, x8 = y8, x9 = y9
       .
       .
       .
       wende (0,1).
       wende (1.0).

       endposition ((1,1,1,1,0,1,1,1,1)).

       poscheck (AP) if gültig (AP,9).
       poscheck (-) if
           beep, nl,
           write ("Eingabefehler."), nl,
           !,
           fail.

       gültig ((),0).
       gültig ((X;T),N) if
          X )= 0,
          X (= 1,
          N1 = N - 1
          gültig (T,N1)

       löse (A,-,Y) if
          endposition (A)
          !,
          Z = ().
          .
          .
          .
goal
       run.
--------------------------------------------------------------------------------
```

56. Prosa

PROgrammiersprache mit **S**ymbolischen **A**dressen.

Prosa ist wie Autocode ein Vorläufer der heute gebräuchlichen Assembler-Sprachen. Wie diese ist Prosa maschinenorientiert. Prosa findet heute kaum noch Anwendung. Gebräuchlich war Prosa zwischen 1966 und 1965.

57. QBE

Query **B**y **E**xample.

Grafische Datenbank-Abfragesprache, von IBM Ende der 70er Jahre im San Jose Research Laboratory entwickelt.

QBE ist non-prozedural (siehe auch SQL).

Im Gegensatz zu SQL ist QBE allerdings nur für große Rechenanlagen tauglich.

58. QL

Query **L**anguage.

Abfragesprache (DL-Sprache).

Query Language hat eine sehr einfache Syntax und Struktur und ist daher leicht erlernbar.

Sie wird hauptsächlich zur Handhabung (Abfragung) von Datenbanksystemen benutzt (siehe auch SQL).

DL-Sprachen sind meist nicht-prozedural. Der Anwender teilt dem System mit, wie das zu lösende Problem aussieht und nicht, wie es zu lösen ist.

59. RPG

Report-**P**rogramm-**G**enerator, auch **LPG** (Listen-Programm-Generator).

RPG und das heute gebräuchlichere RPG II, eine Weiterentwicklung, sind eigentlich keine echten höheren Programmiersprachen im eigentlichen Sinne, sondern, wie der Name schon sagt, Programm-Generatoren, also Systeme, die von Dateien eingelesene Daten-

AUSWERTUNG VON DATENSÄTZEN

sätze nach vorherbestimmten, gleichartigen Umformungen bearbeiten und meistens in Form einer Liste (daher LPG) ausgeben. RPG findet deshalb hauptsächlich in der kommerziellen Datenverarbeitung Anwendung, wo solche Probleme überwiegend auftreten.

Man spricht bei RPG von einer parametrischen Programmierung, da alle Anweisungen und Angaben in Form von Parametern beschrieben werden. Mit Hilfe dieser Parameter wird das Programm dann erstellt.

Erst die Entwicklung von RPG II konnte eine vollständige, gute und umfassende Bearbeitung aller, im kommerziellen Bereich auftretenden Probleme bewältigen. Dennoch kann RPG nicht zu den problemorientierten Programmiersprachen gezählt werden, obwohl einige Softwarehäuser dies gerne sähen. RPG enthält keine frei kombinierbaren Befehle. Und dies ist nun einmal eines der Kriterien für eine problemorientierte Programmiersprache.

RPG wurde ursprünglich von IBM entwickelt und von verschiedenen anderen Softwareherstellern weiterentwickelt. RPG II

wird meist in die Standard-Sprachumgebung des Rechners eingegliedert. Programme in RPG II können Module in anderen Sprachen aufrufen sowie diverse Laufzeit-Bibliotheksprogramme. Andererseits werden die Ergebnisse der RPG II-Programme (Listen) so ausgegeben, daß sie ohne weiteres auch von anderen Programmen in anderen Sprachen weiterverarbeitet werden können.

60. RPNL
Reverse **P**olish **N**otation **L**anguage.
RPNL ist eine höhere Programmiersprache, die sowohl mit Pascal als auch mit Forth große Ähnlichkeiten aufweist. Der erste Unterschied zu anderen Programmiersprachen ist die Verarbeitung der „Umgekehrten Polnischen Notation" (UPN) bei den Operationen (siehe hierzu auch EOL). RPNL unterstützt die strukturierte Programmierung und läßt hohe Ausführungsgeschwindigkeiten zu. RPNL ist compilerorientiert, obwohl das vom Compiler erzeugte Objektprogramm noch keinen direkten Maschinencode darstellt. Man spricht hierbei von einem „Zwischencode". Dieser Zwischencode besteht aus einer Zusammenstellung von Bibliotheksprogramm-Aufrufen, die allerdings sehr maschinennah sind, was sich positiv auf die Ausführungsgeschwindigkeit auswirkt.
In RPNL werden grundsätzlich nur zwei Datentypen unterschieden:
■ *Numerische Daten:*
INTEGER-TYP
(bis zur Größe 2 hoch 16 –1),
BOOLEAN-TYP
(True, False),
ARRAY-TYP
(Zusammenfassung verschiedener oder gleicher Typen).
■ *Alphanumerische Daten:*
STRING
(maximal 255 Zeichen lang).

Alle in einem RPNL-Programm verwendeten Variablen oder Konstanten müssen am Programmanfang definiert werden. Folgende Namen dürfen nicht als Variablen oder Konstanten benannt werden, da sie bereits vergeben sind:

■ **I, J, K**
als Laufindizes in Schleifen,
■ **TRUE, FALSE**
als logische Werte für „wahr" oder „falsch".

Tabelle 20:

Wahrheitstabelle der Boolschen Operatoren (Prädikate)

Operand 1	Prädikat	Operand 2	Ergebnis
TRUE	NOT	-	FALSE
FALSE	NOT	-	TRUE
TRUE	AND	TRUE	TRUE
TRUE	AND	FALSE	FALSE
FALSE	AND	TRUE	FALSE
FALSE	AND	FALSE	FALSE
TRUE	OR	TRUE	TRUE
TRUE	OR	FALSE	TRUE
FALSE	OR	TRUE	TRUE
FALSE	OR	FALSE	FALSE

Der Programmaufbau eines RPNL-Programmes ist grundsätzlich in drei Teile gegliedert:

1. Programmklammer mit Namen
Die erste Zeile eines RPNL-Programmes enthält immer die Anweisung:
PROGRAMM Programmname
In der letzten Zeile steht immer . . . END

2. Deklarationsliste
Beispielsweise VAR für Variablenname,
CONST für Konstantenname,
Anzahl ARRAY Feldname.

Hier wird einerseits der Variablen/Konstanten- und Feldname deklariert, andererseits Speicherplatz zugewiesen und belegt: normalerweise für jede Größe 16 Bit, oder für die Größe des Feldes (Anzahl) * 16 Bit.

3. Anweisungsliste
Die Anweisungsliste enthält die Algorithmen, welche den Lösungsweg des Problems beschreiben.
Man unterscheidet bei RPNL drei verschiedene Anweisungstypen:

a) einfache Anweisungen,
b) E/A-Anweisungen für die Ein- und Ausgabe,
c) Steueranweisungen für Datenfluß und Ausführungssteuerung.
Einfache Anweisungen können aus einstelligen oder zweistelligen Operatoren bestehen. Einstellige Operatoren bearbeiten einen Operanden, zweistellige Operatoren benötigen zwei Operanden.

Einstellige Operatoren:
INC bildet das Inkrement des angegebenen Operanden (=Erhöhung um 1),
DEC das Dekrement wird gebildet (Verringerum um 1),
NOT logische Verneinung (Negation),
DUP Duplizierung eines Wertes (Kopie),
? Abruf eines Speicherplatzes, einer Variablen oder Konstanten.

Zweistellige Operatoren:
\+ Addition,
– Subtraktion,
DIV Division (nur ganzzahlig), der Rest ist verloren,
* Multiplikation,
MOD Division zweier Operanden, welche nur den errechneten Rest anzeigt (Modulo Division),
SWAP Vertauschung von zwei Operanden,
AND Logisches UND (Konjunktion),
OR Logisches ODER (Disjunktion).

Zur „Wahrheitstabelle der Boole'schen Operatoren" siehe *Tabelle 20.*

Außerdem stehen noch die logischen Vergleichsoperatoren gleich/größer/kleiner und deren Kombination zur Verfügung. Sie liefern für Integer-Operanden logische Wahrheitswerte als Ergebnis.

Ein- und Ausgabeanweisungen und RPNL-Prozeduren:
CR Carriage Return (Zeilenvorschub)
WRITE Ausgabe von Text (in Hochkommas),
PRINT Ausgabe einer Integerzahl,
PRINT(S) Ausgabe eines Strings,
READ liest eine Integerzahl ein,
READ(S) liest einen String ein,
STRING weist einem Text Speicherplatz zu.

Bedingungen und Schleifen
IF-THEN-ELSE
Vor dem Schlüsselwort IF steht ein Ausdruck, der die Bedingung darstellt (ein logischer Wert), da-

hinter die Anweisungsliste, die nur dann ausgeführt wird, wenn die Bedingung erfüllt ist (TRUE). Die Anweisungsliste hinter ELSE wird genau dann ausgeführt, wenn die Bedingung vor dem IF nicht erfüllt ist (FALSE).

REPEAT UNTIL LOOP
Hinter REPEAT folgt die Anweisungsliste, deren Anweisungen wiederholt werden sollen. Abgeschlossen wird die Schleife durch das Schlüsselwort LOOP. An einer geeigneten Stelle zwischen REPEAT und LOOP steht UNTIL, dem direkt ein Ausdruck (logischer Wert) vorangeht.
Abhängig von dem aktuellen Wert des Ausdruckes erfolgt die Fortsetzung des Programms mit der ersten Anweisung hinter UNTIL (bei FALSE) oder hinter LOOP (bei TRUE) und damit wird aus der Schleife herausgesprungen.

FOR LOOP
Vor dem Schlüsselwort FOR müssen zwei Operanden angegeben werden, die den Endwert und den Startwert der Laufvariablen festlegen. Zwischen FOR und LOOP steht die Anweisungsliste. Es können drei Schleifen ineinander verschachtelt werden. Die Variable I ist immer für die innere Schleife reserviert, J für die mittlere und K für die äußere.
Bei nur einer Schleife oder einer Verschachtelung von zwei Schleifen gilt das oben Gesagte analog.
(Programmbeispiel siehe Tabelle 21.)

61. SIMULA
SIMULA ist mehr ein System als eine Sprache. Mit Hilfe des SIMULA-Systems lassen sich beispielsweise rechnerinterne Abläufe anhand von Modellen untersuchen. So wurden umfangreiche Untersuchungen von Betriebssystemen mit Hilfe von SIMULA durchgeführt.
SIMULA gibt es seit etwa 1965.
Der Vorteil gegenüber anderen Programmiersprachen liegt in der Struktur des Systems, welches sich besonders zu Simulationszwecken eignet (daher der Name SIMULA).
Es gibt verschiedene SIMULA-Systems, auf die aber wegen ihrer Spezialisierung hier nicht eingegangen werden kann.

```
Tabelle 21:
------------------------------------------------------------
Programmbeispiel zu RPNL
Listing
------------------------------------------------------------
PROGRAMM BEISPIEL
   5 ARRAY LISTE
   VAR ZAHL
   4  0  FOR
      REPEAT
      WRITE "EINGABE" CR
         ZAHL READ
         ZAHL ? I 2 * LISTE + ! =
      ZAHL ? 100 (= UNTIL
         ZAHL PRINT
         WRITE "ZU GROSS" CR
         LOOP
      LOOP
END
------------------------------------------------------------
```

62. SIL 3
System Implementation Language 3.
Maschinenunabhängige System-Implementierungssprache (SIL).
Sie dient zur Software-Herstellung und zeichnet sich in erster Linie durch ihre Codeeffizienz aus.
Die Programmierung von SIL ist recht kompliziert, nicht zuletzt durch die fehlende Problemnähe und die speziellen Anforderungen an der Anwendersoftware.

63. Smalltalk 80
Smalltalk wurde Anfang der siebziger Jahre entworfen, 1980 fertiggestellt (daher Smalltalk 80) und 1981 zum ersten Mal der Öffentlichkeit vorgestellt. Im „Xerox Paolo Alto Research Center" entwickelt, ist Smalltalk zugleich Programmiersprache und Rechensystem. Großen Wert bei der Ausstattung dieses Systems wurde auf grafische Interaktionsmechanismen wie Windows gelegt. Menü- und Maustechniken werden ebenfalls unterstützt. In Smalltalk konnte also schon das realisiert werden, was später von Betriebssystemen (grafischen Benutzeroberflächen) wie GEM oder anderen übernommen wurde.
Das Ausgabemedium für Smalltalk ist immer ein Bitmap-Display, da jede Ausgabe grundsätzlich grafisch erfolgt. Dies geschieht mit Hilfe sogenannter „Forms". Cursor, Bildfläche und auch Schriftarten werden alle als Forms gehandhabt. Smalltalk enthält zwei Editoren zur Veränderung der Forms.
Auch echte Grafik ist in Smalltalk möglich: Liniengrafik und Splines, Kreise sowie andere geometrische Figuren.
Zur Simulation gibt es in Smalltalk das sogenannte Klassenkonzept, welches von der Programmiersprache SIMULA übernommen wurde. Die Klassen dienen den Wahrscheinlichkeitsverteilungen und ereignisgetriebenen Simulationen.
Smalltalk ist, wie schon erwähnt, nicht nur eine Sprache, sondern gleichzeitig ein System. Das System ist seinerseits in Smalltalk geschrieben und basiert auf einer virtuellen Maschine. Das System ist dem Benutzer zugänglich und ermöglicht so Eingriffe.
Für alle Anwenderprogramme wird ein einheitlicher Mechanismus für die Realisierung der interaktiven Schnittstelle zur Verfügung gestellt – ein ähnliches Vorgehen wie bei den Betriebssystemen mit grafischer Benutzeroberfläche. Dieser Mechanismus besteht aus den Bestandteilen Model, View und Controller. Das Modell wird von den Daten und Informationen gebildet. Die View beschreibt die Darstellungen und der Controller übernimmt die Steuerung, also er beschreibt, welche Auswirkungen Ereignisse (etwa Benutzereingaben) auf das System haben.

SPRACHE UND SYSTEM

Smalltalk ist objektorientiert. Jede Komponente von diesem System ist ein Objekt. Die Programme in Smalltalk beschreiben lediglich die Kommunikation dieser Objekte untereinander und natürlich die Objekte selbst.
Die Programmierung in Smalltalk besteht daher im Wesentlichen aus der Neubildung von Objekten und dem „Senden von Botschaften" zwischen diesen.
Die Objekte in Smalltalk 80 sind sehr abstrakt und haben kaum noch etwas mit den technischen Eigenschaften der Hardware des Rechners zu tun oder den klassischen Objekten in der Software, also Zahlen oder Zeichen.
Smalltalk gibt es für ver-

schiedene Rechnersysteme, in der Regel sind dies aber keine Microcomputer. Lediglich für den Macintosh von Apple gibt es eine Implementation.

64. SQL
Structured Query Language.
Strukturierte Abfrage-Sprache, vorwiegend für Datenbanken entwickelt von IBM im San Jose Research Laboratory.
SQL ist seit Ende der 70er Jahre auf dem Markt und wurde seitdem in verschiedensten Datenbanksystemen installiert. Bekanntestes Beispiel ist die relationale Datenbank ORACLE (1979).
SQL ist non-prozedural, also muß nicht der Lösungsweg vom Programmierer beschrieben werden, sondern nur das Problem selbst, das Programm sucht nach Lösungen.
Datenstrukturen können in Form von Tabellen dynamisch definiert und manipuliert werden. Diese Möglichkeiten stehen dem Benutzer über ein spezielles Interface mit dem bezeichnenden Namen UFI = User Friendly Interface zur Verfügung, welches die Datenmanipulation im Dialogbetrieb zuläßt.
SQL besitzt erstaunliche Abfragemöglichkeiten, die jedoch je nach gewünschter Abfrage Zeit kosten können.
SQL bietet hierfür die Möglichkeit, Indizes zur Retrieval-Beschleunigung zu definieren. Die Indexdefinition kann auch dazu benutzt werden, um eine von SQL nicht abgedeckte, jedoch gewünschte Leistung sicherzustellen.
Neben SQL hat sich – ebenfalls von IBM entwikkelt – eine weitere Abfrage-Sprache etabliert: QBE (Query By Example) – mehr ein grafisches Abfragesystem. QBE, auch als Ergänzung zu SQL, ist nur auf mittleren und großen Rechenanlagen einsatzfähig.
Für Mini- und Microcomputer (so VAX von DEC und IBM, PC, XT, AT) gibt es SQL ebenfalls in Verbindung mit der Datenbank ORACLE.

65. SYMAP
Eigenentwicklung der DDR, aufgebaut aus verschiedenen Elementen von Fortran und Exapt.

66. V.I.P.
Visuelle Interaktive Programmiersprache.
Programme in V.I.P. werden über Flußdiagramme eingegeben. V.I.P. ist auf den Apple-Macintosh zugeschnitten und erlaubt Zugriffe auf die Macintosh-Toolbox. V.I.P. verfügt über einen visuellen Debugger und einen sehr schnellen Interpreter. Es gibt auch Transferprogramme, die den V.I.P.-Code in LS-Pascal übersetzen.

67. PL/PC
Programming Language for Personal Computers.
PL/PC ist eine relativ neue Universal-Programmiersprache, speziell geschaffen zum Einsatz auf PCs unter MS-DOS.
Wie BASIC ist PL/PC interaktiv, hat aber diesem gegenüber den Vorteil, strukturiert zu sein (wie MODULA-2).
Auch von APL sind Elemente bei der Schaffung dieser neuen Programmiersprache übernommen worden; so zum Beispiel die mächtigen Datenstrukturen und Manipulationsfunktionen.
Zum Sprachumfang gehören Editoren für Datenstrukturen, die wie ein Spreadsheet aufgebaut sind, und Text zur Deklaration von Subroutinen, Compiler für selbst definierte Subroutinen, Interpreter, Debugger, Grafikunterstützung für alle bekannten Bildschirme einschließlich Turtle-Grafik.
Für rechenintensive Anwendungen existiert eine Spezialversion, welche die Coprozessoren 8087 und 80287 unterstützt.
Variable können ihren Typ (Boolean, Byte, Integer, Long [32 Bit], Real und Complex, String, Charakter) dynamisch ändern.
Die meisten mathematischen Funktionen sind eingebaut.
PL/PC ist leicht lernbar und daher auch für Anfänger geeignet, hat aber gegenüber BASIC gewisse Vorteile.
Am Ende dieses Beitrages können wir nur hoffen, etwas Klarheit in dieses Sprachengewirr gebracht zu haben.

Ende der Serie

Oliver Rosenbaum □

IMPRESSUM C16/P4 SPECIAL

erscheint zweimonatlich in der CA-Verlags GmbH, einer Gesellschaft der Aktuell-Gruppe

VERANTWORTLICH FÜR DEN INHALT:
Anton Kult
Alfons Mittelmeyer

REDAKTION UND STÄNDIGE MITARBEITER:
Peter Basch, Harald Beiler, Rosemarie Huber, Lothar Miedel, Michael Reppisch, Rudolf Schmid-Fabian, Torsten Seibt, Hermann Wellesen, Bernd Welte

GESCHÄFTSFÜHRER:
Werner E. Seibt

ANSCHRIFT FÜR ALLE VERANTWORTLICHEN:
Postfach 1161,
8044 Unterschleißheim
Tel.: 089/1298011
Telex: 5214428 cav-d

ANZEIGENVERWALTUNG:
ADV-Mediendienste,
Aindlingerstr. 17–19,
8900 Augsburg 1
Tel.: 08121/7904-227
Telex: 533502
Teletex: 821887
Telefax: 0821/7904-243

VERANTWORTLICH FÜR DEN ANZEIGENINHALT:
Brigitte Kostić
Es gilt Preisliste Nr. 9 vom 1.6.1988
Media-Unterlagen bitte anfordern.

VERTRIEB:
Verlagsunion Wiesbaden

Printed in Germany by
ADV, Aindlingerstr. 17–19,
8900 Augsburg 1

Veröffentlichung gem. §8,3 Bayr.PrG: Gesellschafter der CAV GmbH sind Henny Rose Seibt, Kauffrau, zu 26 v.H. und Werner E. Seibt, Journalist, beide Elisabethstraße 1, 8044 Unterschleißheim, zu 74 v.H.

SERIELL UNGLEICH SERIELL

Wie aus einigen Artikeln von Ihnen über Drucker in der C-WELT hervorgeht, sind Sie wohl einer der besten C-16-Drucker-Fachmänner, die es gibt. Daher bin ich voller Hoffnung, daß Sie mir bei meinem Problem weiterhelfen können. Ich besitze einen IBM-Drucker, den ich bisher erfolglos an meinen C16 anzuschließen versuchte. Er hat genau wie der C16 eine serielle Schnittstelle und sollte eigentlich ohne Interface anzuschließen sein. Allerdings komme ich mit der Steckerbelegung nicht ganz klar. Am Druckerausgang sind nämlich nur drei verschiedenfarbene Kabel. Am Stecker sind noch zusätzlich zwei Kabel, die untereinander im Stecker je zwei Pins miteinander verbinden.
Daraus schließe ich, daß beim seriellen Port des C16 auch Pins miteinander verbunden werden müssen. Leider weiß ich nicht, welche. Da mir an meinem Rechner sehr viel liegt, möchte ich nicht alle Möglichkeiten ausprobieren. Ich habe auch nicht die Gelegenheit, irgendwo so einen Stecker auseinander zu nehmen, um nachzuschauen. Daher wende ich mich hoffnungsvoll an Sie.
Peter Ochs, Biedenkopf

Die seriellen Schnittstellen, die Sie in Ihrem Brief ansprechen, lassen sich, obwohl sie die gleiche Bezeichnung tragen, leider nicht miteinander vergleichen. Die serielle Schnittstelle Ihres Rechners entspricht nicht der allgemeinen Norm und ist nur eine Abwandlung des IEC-Bus 625, wie er in der Industrie eine Einigung erzielte. Commodore hat diesen Bus abgemagert und ihn in die Home-Computer VC20 bis PC128 eingebaut. Die Schnittstelle Ihres Druckers aber entspricht der RS-232-Norm, wie sie bei den PCs Einzug fand.
Wie Sie sehen, handelt es sich damit um zwei grundverschiedene Ausführungen von Schnittstellen, bei denen außer den mechanischen Anschlüssen auch noch die elektrischen Werte nicht übereinstimmen. Sie benötigen für Ihren Drucker ein serielles Interface. Leider gibt es auch hierbei immer Probleme, weil oft die Leitung umzulöten sind und das Handshaking sich von Drucker zu Drucker unterscheiden kann. Wenden Sie sich am besten an eine Firma, die Interfaces herstellt und Ihnen somit weiterhelfen kann. Einen Katalog mit Interfaces können Sie kostenlos beziehen bei:
Wiesemann & Theiss GmbH
Abteilung: CW
Winchenbachstraße 3–5,
D-5600 Wuppertal 2

FALSCHE PIN-BELEGUNG AM CENTRONIC-INTERFACE

Am liebsten hätte ich laut „Hurra" geschrien, als ich Ihren Artikel las, denn ich bin seit zwei Wochen stolzer(?) Besitzer eines Druckers „Star LC-10 multi-font". Abgesehen davon, daß man mich beim Kauf regelrecht „abgelinkt" hat – es wurde mir auf ausdrückliches Befragen zugesichert, der Drucker würde am Plus4 laufen, auf meine Reklamation wollte man davon natürlich nichts mehr wissen –, habe ich mir die Hacken abgelaufen nach einem der von Ihnen genannten Interfaces. Aber es gibt keines! Angeblich sei der Drucker noch so neu, daß es noch keines gäbe. Ich habe aber eher den Eindruck, als ob man sich in den Computerläden nicht so besonders für Kundennöte interessieren würde.
Um so erfreuter war ich, Ihren Artikel zu finden. Also ab ins Elektronik-Geschäft, die Bauteile gekauft und ein Wochenende gelötet. Dann das Programm (dank Checksummer ein Klacks) eingegeben, alles miteinander verbunden und ..... nichts!
Der Drucker machte keinen Mucks, ganz abgesehen davon, daß es mir nicht einmal gelingt, Daten an ihn zu senden.
Ich gehe folgendermaßen vor: Programm „Centronics" laden – Run. Der Plus4 meldet „Ready". Danach öffne ich den Druckerkanal mit „open,4", wiederum meldet sich der Computer mit „Ready". Wenn ich jetzt cmd1 eingebe, verschwindet der Cursor auf Nimmerwiedersehen und läßt sich auch nicht mehr hervorzaubern. Der Rechner spricht auf keine Taste mehr an, auch nicht auf „Help". Es bleibt in diesem Fall nur noch ein „Reset", um den ganzen Tanz von vorne zu beginnen. Aber drucken, das will er nicht.
Für mich als Laien gibt es nur zwei Erklärungen: Entweder Ihr Artikel stimmt doch nicht ganz (er müßte dann heißen „... mit jedem Drucker außer dem LC-10), oder aber der Fehler liegt bei mir. Dann kann es aber nur mein Interface sein. Aber ich bin der Meinung, alles richtig gemacht zu haben und außerdem habe ich es mehrfach durchgemessen.
Können Sie mir in irgendeiner Weise helfen? Ich lege Ihnen einen Belegplan des Centronics-Eingangs meines Druckers bei, eventuell nützt Ihnen der etwas.
In diesem Zusammenhang noch eine Frage: In der Abbildung des Centronics-Steckers in Ihrem Artikel finden sich auf pin 19 ff insgesamt 12 Leitungen, die jedoch in Ihrem Text in keiner Weise genannt sind. Deshalb habe ich sie auch nicht angelötet, ganz zu schweigen davon, daß ich gar nicht wüßte, wohin mit dem anderen Ende. Ist das der Fehler?
Rolf-Peter Schulz, Stuttgart

In der Interface-Beschreibung war ein Fehler enthalten. Die Strobe-Leitung liegt am Centronics-Port auf Pin eins und die Busy-Leitung auf Pin elf.
In dem Artikel „Drucken mit jedem Modell" stand infolge einer Verwechslung das Gegenteil zu lesen. Demnach sind im Heft C16/P4-SPECIAL auf Seite 10 in der rechten unteren Ecke der „Anschlußtabelle" die beiden untereinanderstehenden Zahlen eins und elf auszutauschen. Wenn Sie entweder am Interface oder am Centroncs-Stekker die beiden Leitungen umlöten, so müßte der Ausdruck funktionieren. Übrigens können Sie auch einen deutschen Zeichensatz mit Ihrem Drucker realisieren. Allerdings ist hierzu das in der COMMODORE-WELT Nr. 12/87 beschriebene Centronics-Kabel vonnöten, das am Userport des Plus4 angeschlossen wird.
Mehrere Leitungen ab Pin 19 wurden früher aus Störungsgründen an Gehäusemasse gelegt. Bei den heutigen Kabeln ist es besser, dies nicht zu tun. Daher sind die Pins 19 bis 30 nicht anzuschließen. Das gleiche gilt ebenso für Pin 17.

DEUTSCHE ZEICHEN FÜR MPS 801 UND MPS 803

In der Nr. 1 dieses Jahrgangs habe ich Ihr Zeichensatz-Programm auf Seite 22 bis 31 mit Begeisterung zur Kenntnis genommen und als Plus4-Besitzer natürlich sofort abgetipt, angepaßt an Drucker und Plus4-Software und in Betrieb genommen.
Dabei habe ich auf dem Bildschirm und auf dem Drucker Commodore MPS 803 die großen und

kleinen Umlaute, den Paragraphen und das „ss". Es tritt nur folgende merkwürdige Erscheinung auf, die ich trotz umfangreicher Bemühungen nicht abstellen konnte: Alle Anfangsbuchstaben und im Text großgeschriebenen Buchstaben werden klein ausgedruckt und alle klein eingegebenen Buchstaben werden als Großbuchstaben ausgedruckt. Ich konnte bisher keine Änderung herbeiführen.
Damit die wesentlichen Parameter überprüft werden können, falls Sie das Problem interessiert, füge ich einen Ausdruck des Druckeranpassungs-Listings bei. Wenn Sie mir helfen könnten, wäre ich Ihnen sehr verbunden.
Im übrigen darf ich Sie zu Ihrer schönen Zeitschrift C16-Plus4 SPEZIAL beglückwünschen, die ich mir stets besorge und die mir schon sehr viel Freude bereitet hat.
Herbert Fiedler
Marktheidenfeld

Wie ich sehe, ist Ihnen die Zeichensatz-Anpassung bis auf eine winzige Kleinigkeit gelungen. In Zeile 240 der Druckeranpassung ist ein c für Commodore-Drucker einzutragen. Wir hatten fälschlicherweise angenommen, der MPS-801 würde auch eine doppelte Grafikdichte besitzen. Sie haben aber bereits den passenden Zeichensatz selbst entwickelt. Den Code zwei, der zum Umschalten in die doppelte Grafikdichte vorgesehen war und in den Code-Sequenzen gleich auf die anfängliche Acht folgt, können Sie weglassen. Beim MPS-801 trat nach einem undefinierten Druckerzeichen vor einem Return das Phänomen auf, daß der Zeilenabstand an dieser Stelle nicht mehr richtig stimmte. Sie können dieses Problem durch eine kleine Änderung beseitigen:

```
650 DATA 0,13,-1
```

Außerdem würde uns sehr interessieren, wie Sie unsere Drucker-Codes für den MPS-801 und MPS-803 finden:

```
550 data 8,160,213,212,
     189,192,128,15,-1
     :rem "ae"
560 data 8,185,196,196,
     185,128,128,15,-1
     :rem "oe"
570 data 8,189,192,192,
     189,192,128,15,-1
     :rem "ue"
580 data 8,248,134,137,
     215,224,128,15,-1
     :rem "ss"
590 data 8,241,172,162,
     172,241,128,15,-1
     :rem "Ae"
600 data 8,184,197,196,
     197,184,128,15,-1
     :rem "Oe"
610 data 8,188,193,192,
     193,188,128,15,-1
     :rem "Ue"
640 data 8,138,213,213,
     213,168,128,15,-1
     :rem paragraph
```

SPEICHERN MIT DEUTSCHEM ZEICHENSATZ UND PLUS4-SOFTWARE

Ich habe mich über Ihr Antwortschreiben zu meinen Problemen hinsichtlich des deutschen Zeichensatzes auf dem Plus4 sehr gefreut, zumal ich nunmehr tatsächlich die Plus4-Software mit deutschem Zeichensatz benützen kann. Auch das Ausdrucken mit dem MPS 803 ist nunmehr problemlos möglich.
Ein kleines Problem habe ich allerdings noch. Nach dem Text in C16-PLUS4-SPEZIAL Nr. 1, Jahrgang 1988, war ich der Meinung, daß mit dem durch ZSKOMP erstellten neuen Programm aus Screen/Keyboard und Druckeranpassung Texte erstellt, abgeändert, abgespeichert, geladen und ausgedruckt werden können.
Ich kann Texte erstellen, abändern, laden und speichern mit dem Teilprogramm Screen/Keyboard. Mit dem mit ZSKOMP erstellten neuen Programm kann ich lediglich Texte laden und drucken. Wenn ich hier ein geladenes Programm abändere, kann ich dieses wohl ändern, aber nicht abspeichern.
Versuche ich das, erscheint auf dem Bildschirm lediglich ein Zeichensalat. Auf der Diskette ist dann wohl der Filename vorhanden, unter dem ich den Text abspeichern wollte. Aber offensichtlich wird der eigentliche Text nicht abgespeichert. Auf der Diskette ist dann auch jeweils nur ein Block durch den Filenamen belegt.
Will ich den Text ausdrukken, muß ich das mit ZSKOMP erstellte Programm laden und starten. Dann kann ich den Text laden und ausdrucken.
Ich kann dann wohl auch noch den Text ändern, ihn aber mit diesem Programm nicht abspeichern. Er wäre dann verloren.
Ihre Drucker-Codes für ä, ö, ü, ß, Ä, Ö, Ü, § finde ich in Ordnung. Dieser Text ist bereits mit ihnen erstellt. Besonders interessant finde ich das ß. Vielleicht sollte man das ö etwas größer gestalten, wie etwa in den Ihnen mitgeteilten Codes.
Herbert Fiedler
Marktheidenfeld

Wenn Sie eine zusätzliche Zeile in die DRUCKER-ANPASSUNG einfügen, so sollte das Speichern mit der Plus4-Software möglich sein:

```
1955 POKEAR+19,168
     :POKEAR+20,255
```

In der Tabellenkalkulation treten beim komprimierten Zeichensatz Probleme auf. Diese können Sie jedoch beseitigen, wenn Sie im Komprimierungs-Programm in Zeile 555 den an dritter Stelle stehenden DATA-Wert 128 durch die Zahl 156 ersetzen.
Das ö könnte etwas größer gestaltet werden. Ein Vorschlag ist, die Codes vom großen Ö für das kleine herzunehmen. Für das große Ö könnte man versuchen:

```
600 DATA 8,189,194,
     194,194,189,128,
     15,-1
```

VC20-TASTATUR FÜR C116 MIT 61 KBYTE

Angeregt durch „Weg mit Gummitasten...", Sonderheft COMMODORE WELT 1/87, Seiten 124 ff, habe ich an meinen C116 eine VC-20-Tastatur angeschlossen.
Fast alles funktioniert – mit einigen unwesentlichen Verschiebungen wie zum Beispiel CLEAR/HOME-Funktion auf Linkspfeil-Taste (direkt über CTRL-Taste). Sehr störend wirkt sich aber der Umstand aus, daß auf der rechten SHIFT-Taste der VC-20-Tastatur die ESCAPE-Funktion des C116 gelandet ist. Das macht, etwa bei Textverarbeitung, den Umbau hinfällig.
Gibt es eine Möglichkeit, die Funktion der rechten SHIFT-Taste herzustellen und die ESCAPE-Funktion zum Beispiel auf die RESTORE-Taste zu verbannen? Linke SHIFT und SHIFT/LOCK funktionieren.
Heinz Wollmann
Umkirch bei Freiburg

Es freut uns, daß Ihnen die Anpassung der VC-20-Tastatur an den C116 beinahe gelungen ist. Da Tastaturen meistens in einer Matrix angeordnet werden, um Verbindungsleitungen zu sparen, ist es sehr schwierig, andere als die vorgesehenen Tastaturen an einen Computer hundertprozentig anzupassen. Für die von Ihnen beschriebene Möglichkeit bleibt wahrscheinlich nur eine Hardwarelösung übrig. Das heißt, Sie müssen die Tastatur öffnen und die Verbindungen vornehmen. Dazu müssen Sie die Leiterbahnen der entsprechenden Tasten direkt neben den Kontaktstellen auftrennen und durch Drähte mit den Kontaktstellen der jeweiligen anderen Taste verbinden. Als einfache Lö-

ung bietet sich auch der Austausch der C116-Tastatur gegen eine Tastatur des Plus4 an, die elektrisch identisch mit der C116-Tastatur ist, aber die besseren Tasten besitzt.

ABGESCHNITTENE STRINGS BEIM INPUT

Beim Ablegen von Strings in sequentielle Dateien habe ich folgendes Problem:
Handelt es sich bei den ersten oder letzten Stellen des Strings um Leerstellen, werden diese nicht in der Datei abgelegt. Dies ist insofern lästig, als sich der Bildschirmaufbau nach Wiedereinlesen und Ausgabe der Strings verschiebt.
Die einzige Möglichkeit, die ich gefunden habe, dieses Problem zu lösen, besteht darin, den String links und rechts um ein beliebiges Zeichen zu ergänzen und diese nach dem Wiedereinlesen zu entfernen.
Sollte Ihnen eine elegantere Methode bekannt sein, wäre ich für eine Mitteilung dankbar.

Josef Lachenmayer
Kaufbeuren

Beim INPUT werden Leerstellen am Anfang und am Ende abgeschnitten. Gegen die abgeschnittenen Leerstellen am Ende läßt sich nichts unternehmen. Das ist aber auch nicht nötig, da Sie den entsprechenden String bei der Ausgabe durch Anfügen einer ausreichenden Menge von Leerzeichen und Formatieren mit der LEFT$-Anweisung auf die richtige Länge bringen können. Um Leerzeichen, Kommas und Doppelpunkt bei einer Eingabe erfassen zu können, ersetzen Sie INPUTA$ durch POKE17,64:SYS 34906:SYS37210A$. Der POKE-Befehl braucht bei mehreren hintereinanderfolgenden modifizierten INPUT-Anweisungen nur einmal am Anfang zu stehen. Er ist gültig bis zur nächsten normalen INPUT-Anweisung.
Zum Wiedereinlesen vom Datenträger – das File muß natürlich bereits mit OPEN geöffnet worden sein – gehen Sie folgendermaßen vor: POKEn Sie die Filenummer in die Speicherstelle 2035 und rufen Sie SYS 65478 auf. Wie CMD den Ausgabekanal öffnet, so öffnet dieser SYS-Aufruf den Eingabekanal. Um dem Rechner mitzuteilen, daß die Eingaben nicht von der Tastatur erfolgen, ist in die Speicherstelle 19 noch eine Zahl ungleich Null zu POKEn. Jetzt können Sie, wie bereits beschrieben, Ihre Daten mit dem modifizierten INPUT vom Datenträger einlesen. Nach den Einlesevorgängen ist mit POKE19,0 dem Rechner mitzuteilen, daß die Tastatur nun wieder als Eingabegerät dient, und mit SYS 65484 sie als Eingabegerät zu definieren.

LESERTIP: DISKETTENBEFEHLE MIT STRING-VARIABLEN

Zum Beitrag „Scratch und Rename im Programm" eine kleine Verbesserung: Die Diskettenbefehle SCRATCH und RENAME funktionieren auch in Verbindung mit Stringvariablen, sofern diese in Klammern gesetzt werden.
Beispiele:

```
SCRATCH(A$)
RENAME(ALT$) TO
        (NEU$)
```

Fehler treten bei mir auf, wenn ich folgenden Befehl eingebe:

```
RENAME(ALT$) TO
        (NEU$),D0,U8
```

Dies sollte jedoch laut Handbuch möglich sein. Dagegen funktioniert einwandfrei:

```
RENAME(ALT$) TO
        (NEU$),U8
```

Florent Teuber

Computer-Typ	BASIC-Anfang	DEZ-Ende	BASIC-Anfang	HEX-Ende	RAM-Byte
C16	4096	– 16383	1000	– 3FFF	12287
C116	4096	– 16383	1000	– 3FFF	12287
Plus4	4096	– 64767	1000	– FCFF	60671
C64	2048	– 40959	0800	– 9FFF	38911
128PC	7168	– 65279	1C00	– FEFF	58111
VC20<3K	4096	– 7679	1000	– 1DFF	3583
VC20+8K	4608	– 16383	1200	– 3FFF	11775
VC20+16K	4608	– 24575	1200	– 5FFF	19967
VC20+24K	4608	– 32767	1200	– 7FFF	28159
PET4K	1024	– 4095	0400	– 0FFF	3071
PET8K	1024	– 8191	0400	– 1FFF	7167
PET16K	1024	– 16383	0400	– 3FFF	15359
PET32K	1024	– 32767	0400	– 7FFF	31743
CBM600	0002	– 65535	0002	– FFFF	65533

Hätten Sie es gewußt?

An dieser Stelle werden wir kleine Hilfen für den täglichen Kampf mit dem Computer veröffentlichen oder „Das weiß doch Jeder"-Themen wiederholen, so daß auch die Einsteiger zu ihrem Recht kommen. Auch Dinge, die bekannt sein sollten oder schon wieder in Vergessenheit geraten sind, werden ausführlich beleuchtet.

Der nutzbare Platz für BASIC ist bei den Commodore-Rechnern unterschiedlich angelegt. Bei BASIC-Programmen, die nicht automatisch an den BASIC-Anfang des Rechners geladen werden, kann es daher Probleme geben. Floppyspeeder laden meistens alle Programme absolut. Wenn Sie zum Beispiel ein vom 128PC abgespeichertes Programm mit dem Plus4 mit LOAD "name",8,1 laden, erscheint es ab der Speicherstelle DEZ 7169.
Der Plus4 benötigt es aber ab DEZ 4097. Laden Sie das Programm jedoch mit LOAD"name",8 , wird es vom Plus4 automatisch an seinen BASIC-Anfang DEZ 4097 gesetzt.
Beim erneuten Abspeichern kann das Programm auch absolut geladen werden, weil es der Plus4 mit seiner BASIC-Anfangsadresse DEZ 4097 abgespeichert hat. Bestimmte Commodore-Computer wie etwa der CBM 8032 laden BASIC-Programme von anderen Rechnern nicht automatisch an ihren BASIC-Anfang. Also suchen sie nach einem Ladeversuch vergeblich.
Der rechnerische Wert der RAM-Bytes kann von der tatsächlichen Systemmeldung abweichen.

Vom Joystick bis Script Plus: Tips & Tricks für C16/P4

Wie gewohnt, bringt die Seite 16 wieder Tips und Tricks, wie Sie Ihren Computer noch besser nutzen können. Diesmal zeigen wir Ihnen, wie man mit Hilfe der beiden Joystick-Ports eine Alarmanlage realisieren kann. Außerdem finden Sie das Hilfsprogramm Silbentrennung zur Bearbeitung von SCRIPT/PLUS-Texten.

PIN-BELEGUNG DES JOYSTICK-PORTS – PORT 1 UND 2 ALS KONTROLLSTATION

Die Funktion eines Joysticks ist denkbar einfach. Die Pin-Belegung ist wie folgt:

Pin	Belegung	
1	oben	8 7 6
2	rechts	
3	unten	
4	links	5 4 3
5	+5V	
6	Feuerknopf	
7	Masse	2 1

Wird nun zum Beispiel Pin 1 (oben) mit Pin 7 (Masse) verbunden, ergibt PRINT JOY(1) die Zahl 1. Im Fachhandel gibt es konfektionierte Kabel mit entsprechendem Stecker. Sicherheitshalber sollte der Anschluß für +5V isoliert werden, damit kein Kurzschluß entsteht. Die anderen Anschlüsse sind in diesem Zusammenhang unproblematisch. Wenn das Geschäft keine passenden Kabel vorrätig hat, kann auch ein Joystick-Adapter verwendet werden, der normalerweise benutzt wird, um Joysticks mit neunpoliger Sub-D-Buchse (C64) an den C16/Plus4 anzuschließen. Hier sind in der Regel nur die sechs für den Joystick notwendigen Kabel angeschlossen. Ein einfacher Test kann mit einem Miniprogramm durchgeführt werden:

```
0 print joy(1):goto
```

Wird nach dem Starten dieser Zeile eines der Kabel von 1 bis 4 oder 6 mit Pin 7 verbunden, ändert sich der ausgedruckte Wert. Übrigens springt ein GOTO ohne Zeilennummer zu Zeile 0.

DER C16 ALS ALARMANLAGE

Die Verbindung der entsprechenden Pins mit Masse (Pin 7) kann auf vielfältige Weise geschehen. Die einfachste Methode wird im Innern des Joystick angewandt. Hier sind einfach kleine Schalter oder Metallkontakte, die die einzelnen Pins auf Masse legen. Genausogut kann man auch sogenannte Reed-Relais verwenden. Diese werden bevorzugt in Alarmanlagen eingesetzt, um das Öffnen und Schließen von Fenstern und Türen zu kontrollieren. Ein Reed-Relais besitzt zwei metallische Kontakte, die sich schließen, wenn ein Magnet in die Nähe gebracht wird und sich öffnet, wenn er wieder entfernt wird. Diese Schaltkontakte sind sehr klein und zudem preiswert zu erhalten. Genauso kann mit einem Quecksilber-Neigungsschalter das Anheben oder Erschüttern eines Gegenstandes kontrolliert werden. In gleicher Weise kann eine Matte mit Trittkontakten eingesetzt werden. Soll die Temperatur eines Gerätes überwacht werden, so kann dafür ein Temperaturschalter verwendet werden. Dieser betätigt sich bei einer bestimmten Temperatur (zum Beispiel 100 Grad) und man kann dies mit dem Joystickport leicht überwachen. In der gleichen Weise kann mit einer Reflexlichtschranke oder einer Gabellichtschranke das Vorhandensein oder Fehler eines Metall- oder Papierbandes überwacht werden. Mit Hilfe einer Taschenlampenbirne oder einer infraroten Leuchtdiode und eines Fototransistors kann eine einfache Lichtschranke gebaut werden. Dazu eignet sich zum Beispiel als Sender der Typ LD261 und als Empfänger BPX81. Der Kontakt ist geschlossen, wenn der Fototransistor beleuchtet wird. Er ist offen, wenn Dunkelheit herrscht. Für eine solche Anwendung haben wir das Beispielprogramm **SCHALTER** geschrieben.
Eine typische Anwendung wäre ein Physikerversuch mit der sogenannten schiefen Ebene. Entlang der Fahrbahn eines kleinen Modellautos befinden sich zehn Fototransistoren, die mit den entsprechenden Pins an den Ports 1 und 2 (mit gemeinsamer Masse) verbunden werden. Auf dem Wagen befindet sich eine kleine Lampe, die die Transistoren beim Vorbeifahren schaltet. Die Schalterstellung wird angezeigt. Die Zeitdifferenz zwischen den Schaltern wird in Zeile 80 bestimmt. Da es nur zehn Werte sind, braucht keine DIM-Anweisung zu erfolgen.
Bei diesem Programm wird vorausgesetzt, daß nur jeweils ein Kontakt geschlossen wird. Für Anwendungen, bei denen zunächst alle Schalter geschlossen sind und nur einer geöffnet wird, muß auf die Speicherstelle $FF08 zurückgegriffen werden.

JOYSTICK-ABFRAGE OHNE JOY

Der entscheidende Nachteil der JOY-Funktion für Meßzwecke ist die Tatsache, daß die Funktion nur vernünftige Werte liefert, wenn die Kontakte eine der mit dem Joystick möglichen Stellungen entspricht. Was vernünftig ist, da zum Beispiel „oben“ und „unten“ gleichzeitig nicht möglich ist. Die Abfrage der Joystickpositionen ohne die JOY-Funktion erfordert leider etwas Übung mit Maschinensprache und Kenntnis der Hardware des C16.
Die vier Leitungen für „oben“, „rechts“, „unten“ und „links“ haben bei Port 1 und Port 2 die gleichen Leitungen und sind mit der Tastaturmatrix verbunden. Die beiden Leitungen für die Feuerknöpfe sind getrennt. Die Verbindung mit der Tastaturmatrix wird offensichtlich, wenn der Joystick mit GETa$ abgefragt wird.
Hierzu ein kleines Beispiel:

```
10 v$="56rdt":rem port 1
20 v$=v$+"34wa":rem port 2
30 geta$:printinstr(v$,a$):goto30
```

Hier entspricht „1" der Position „oben" bei Port 1 und „6" der Position „oben" bei Port 2. Die Feuertaste von Port 2 kann so nicht abgefragt werden.
Umgekehrt erhält man die richtigen Werte mit JOY(1) und JOY(2) durch die Tasten 1, 2, HOME, CTRL und SPACE(=Feuer). Ein Joystickspiel, das mit der JOY-Funktion arbeitet, kann also ohne Joystick gespielt werden. Nicht so Maschinenprogramme. Hier erfolgt die Abfrage komplizierter. Wir haben für unsere Leser daher ein kleines Maschinenprogramm geschrieben, das die Joystickposition abfragt:

```
. 065e  78        sei
. 065f  a2 fa     ldx #$fa
. 0661  8e 08 ff  stx $ff08
. 0664  ad 08 ff  lda $ff08
. 0667  8e 08 ff  stx $ff08
. 066a  cd 08 ff  cmp $ff08
. 066d  d0 f2     bne $0661
. 066f  49 ff     eor #$ff
. 0671  58        cli
. 0672  60        rts
. 0673  78        sei
. 0674  a2 fd     ldx #$fd
. 0676  8e 08 ff  stx $ff08
. 0679  ad 08 ff  lda $ff08
. 067c  8e 08 ff  stx $ff08
. 067f  cd 08 ff  cmp $ff08
. 0682  d0 f2     bne $0676
. 0684  49 ff     eor #$ff
. 0686  58        cli
. 0687  60        rts
```

Dies sind zwei Routinen. Die erste (für Port 1) beginnt bei $0065e und wird mit SYS1630 aufgerufen, die zweite beginnt bei $0673 und wird mit SYS1651 gestartet. Man kann sich die Tipparbeit erleichtern, indem man mit dem Transfer-Befehl einen Teil der JOY-Routine im Betriebssystem (ab $BFC1) kopiert:

T BFCD BFDD 0660

Nach dem Aufruf der Routinen steht das jeweilige Ergebnis im Akku-Register und kann durch PEEK(2034) abgefragt werden.

Beispiel:

```
10 sys1630:b=peek(2034):rem port1
20 print
band1;band2;band4;band8;band64;
30 sys1651:b=peek(2034):rem port2
40 print chr$(18)band1;band2;band4;band
8;band128:goto10
```

Mit Hilfe von IF-Abfragen und der AND-Funktion kann jede Schalterstellung erfragt werden:

Beispiel:

```
100 sys1630:b=peek(2034)
110 if band1 then print"oben"
120 if band2 then print"unten"
130 if band4 then print"links"
140 if band8 then print"rechts"
150 goto100
```

PORT-AUSWAHL DURCH SCHREIBZUGRIFF

Wie weiter oben erwähnt wurde, haben die Pins von Port 1 und Port 2 teilweise die gleichen Leitungen. Wie kann dann unterschieden werden, von welchem Port das Signal kommt? Dies geschieht durch einen Schreibzugriff (STX $....) auf $FF08. Je nachdem, welcher Wert gePOKEd wird, liegt die Masseleitung von Port 1 oder Port 2 an Masse. Die beiden möglichen Werte sind:

Port 1:
$FA = 250 = 11111010
Port 2:
$FD = 253 = 11111101

Der Schreibzugriff wird zur Sicherheit wiederholt, um kurzfristige Schwankungen auszugleichen. Die Abfrage geht auch in BASIC:

```
10 a=65288
20 pokea,250:b=peek(a):rem port 1
30 print
band1;band2;band4;band8;band64;band128:
goto 20
```

Die Ergebnisse sind jedoch fehlerhaft, da manchmal der Interrupt zwischen POKE und PEEK den Wert verändert.
Wenn der C16/Plus4 ein Ereignis an einem der Joystickports feststellt, muß dieses Ergebnis in der Regel zu einem Alarmsignal verarbeitet werden. Die einfachste Möglichkeit besteht in der Verwendung des Sound-Befehls und der entsprechenden Bildschirmanzeige.

AUSLÖSEN DES ALARMS

Jedoch kann meist der Fernseher nicht die ganze Zeit eingeschaltet bleiben. Daher hier ein einfacher Tip, wie ein Alarm ausgelöst werden kann. Am Kassettenport stehen fünf Volt zur Verfügung, mit denen normalerweise der Recordermotor betrieben wird. Wird hier ein Piezosummer (verbraucht sehr wenig Strom) angeschlossen, so kann durch Einschalten des Kassettenrecorders dieser Summer betätigt werden. Genauso kann ein Relais gesteuert werden, das eine Sirene in Betrieb setzt.

Wir hoffen, daß Sie durch diese Beispiele angeregt wurden, Ihren C16/Plus4 zu Meßzwecken einzusetzen. Wir würden uns freuen, wenn Sie uns Ihre Anwendungen mitteilen würden.

TEXTE RECHTSBÜNDIG AUSDRUCKEN

Mit SCRIPT/PLUS kann man auf einfache Weise beliebige Texte erstellen. Besonders schön werden diese mit der Formatan-

weisung **jul** (justification ein). Diese bewirkt, daß alle Zeilen gleich lang werden. Fehlende Buchstaben werden durch Leerzeichen zwischen den Worten aufgefüllt. Damit dabei nicht zu große Lükken entstehen, muß am Zeilenende richtig getrennt werden.
Hierzu hat SCRIPT/PLUS die Möglichkeit, sogenannte Soft Hyphen (weiche Trennungszeichen) einzufügen. Dies geschieht mit ESC und COMMODORE/–. Im Handbuch ist

TRENNVORSCHLÄGE

hier ein Druckfehler und es wird die Ctrl-Taste angegeben. Steht ein solches Zeichen im Wort, so wird an dieser Stelle getrennt, wenn sie am Zeilenende zu stehen kommt. Der Nachteil liegt darin, daß nur ein solches Trennungszeichen eingesetzt werden kann. Mehrere Trennvorschläge schaden zwar nichts, da sie normalerweise nicht ausgedruckt werden, aber es wird prinzipiell das erste verwendet, auch wenn ein anderes eine günstigere Trennung ergäbe. Soll ein wichtiger Text im Blocksatz (beispielsweise rechtsbündig) gedruckt werden, ist es unumgänglich, die Trennungen von Hand einzufügen. Dazu muß die Zeilenlänge im Eingangsmenü auf die gewünschte Länge eingestellt werden und nach jeder Silbentrennung der Abschnitt neu formatiert werden. Dies ist sehr aufwendig. Daher haben wir Ihnen das Hilfsprogramm **SILBENTRENNUNG** geschrieben, mit dem die optimale Trennung auf einfache Weise möglich ist. Das Programm ist auf Diskettenbetrieb ausgelegt, kann aber leicht mit Hilfe des Handbuchs für Kassettenbetrieb umgeschrieben werden. Der mit SCRIPT/PLUS (oder einem anderen Programm, das mit ASCII-Zeichen arbeitet) erstellte Text wird in den Speicher geladen und Zeile für Zeile abgearbeitet. Immer, wenn ein Leerzeichen oder Trennungszeichen (auch Soft Hyphen) gefunden wird, überprüft unser Programm, ob die erforderliche Zeilenlänge erreicht ist. Bei einer zu langen Zeile wird diese am Schirm angezeigt und mit Hilfe der Cursortasten (links und rechts) das Trennungszeichen an die gewünschte Position gebracht. Mit Return wird die Zeile abgeschlossen und das Programm fügt automatisch den Return-Code 13 in den Text ein.

PROGRAMM-ERKLÄRUNG

Zeile 100: Der Speicher wird begrenzt, um einen geschützten Textspeicher zu schaffen.
Zeile 160–180: Die Zeilenparameter und der Filename werden abgefragt. Die Variable DM gibt an, wieviele Zeichen noch bis zum Zeilenende fehlen dürfen, damit das Programm automatisch ein Return setzt.
Zeile 200–250: Einlesen des ASCII-Textes in den Speicher. Die Print-Anweisung in 230 kann weggelassen werden (der Ablauf wird dadurch schneller).
Zeile 280: Das alte Textfile wird in eine Sicherheitskopie umbenannt. Hier kann der Rename-Befehl nicht verwendet werden, da dieser keine Variablen verarbeitet. *(Anmerkung der Redaktion:* Der Rename-Befehl kann Variablen verarbeiten, wenn diese in Klammern eingeschlossen werden.)
Zeile 300: Seq-File zum Schreiben öffnen.
Zeile 330–370: Überprüfung, ob ein Trennungszeichen gefunden wurde.
Zeile 380–390: Prüfung auf Zeilenende. Ist die maximale Zeilenlänge erreicht und das letzte Trennungszeichen zu weit ent-

SILBENTRENNUNG

```
10 rem silbentrennung =========c16 <kg>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by  r. schmid-fabian    <jp>
50 rem               heidelberg    <kp>
60 rem                             <ah>
70 rem basic v3.5                  <nl>
80 rem c16/116/p4   floppy         <ba>
90 rem ========================== <jg>
100 poke55,0:poke56,39:clr         <bi>
110 l$=" ":b$="-":rem trennzeichen <le>
120 sh$=chr$(192):rem soft-trennze
ichen                              <li>
130 cr$=chr$(013):cl$=chr$(147)    <bm>
140 li$=chr$(157):re$=chr$(29)     <el>
150 ro$=chr$(146):rv$=chr$(18):d$=
l$+l$+l$+l$                        <pb>
160 input"max. Zeilenlaenge";ml    <df>
170 input"max. Abweichung";dm      <ae>
180 input"file-Name";p$            <nn>
190 rem ****** einlesen ********** <la>
200 trap250                        <kc>
210 open2,8,2,p$+",s,r"            <de>
220 fori=12000to60000              <kj>
230 get#2,a$:pokei,asc(a$):printa$
;                                  <bo>
240 if st=0 then next              <mi>
250 close2:hlt=i                   <hd>
260 rem **** rename v. p$  ******* <el>
270 printchr$(147)"Jetzt wird "+ch
r$(34)+p$+chr$(34)+" umbenannt (Ta
ste)":getkeya$                     <na>
280 open1,8,15,"r:"+left$(p$+d$+d$
+d$,12)+".bak+"+"="+p$:close1      <dl>
290 rem ****** bearbeiten ******** <fc>
300 open2,8,2,p$+",s,w"            <kp>
310 for i=12000 to hlt             <gm>
320 a$=chr$(peek(i))               <gp>
330 if a$=sh$ then tp=zl:en$=b$:go
to380                              <nb>
340 if a$=cr$ then printrv$w$:prin
t#2,w$:zl=0:w$="":goto440          <jh>
350 w$=w$+a$:zl=zl+1               <gk>
360 if a$=l$ then tp=zl:en$=""     <bf>
370 if a$=b$ then tp=zl:en$=b$     <je>
380 if zl<ml then 440              <oa>
390 if(ml-tp)>dm then gosub 480    <na>
400 pr$=left$(w$,tp)+en$:printrv$p
r$                                 <ph>
410 print#2,pr$                    <ig>
420 w$=right$(w$,ml-tp)            <gk>
430 tp=0:zl=len(w$)                <na>
440 next                           <pf>
450 close2:end                     <pg>
460 rem ****** trennung ********** <oo>
470 rem ****** von hand ********** <og>
```

```
480 d=0:x$="":forv=i+1toi+6:x$=x$+
chr$(peek(v)):nextv                <gh>
490 if asc(x$)=32 then i=i+1:tp=le
n(w$):en$="":return                <np>
500 x$=rv$+x$                      <pn>
510 pr$=right$(w$,ml):z$=left$(pr$
,ml-d):t$=right$(pr$,d)            <lo>
520 en$=b$:p=peek(i-d):if p=32 the
n en$=l$                           <le>
530 printchr$(147)+z$+rv$+en$+ro$+
t$+x$                              <al>
540 getkeyq$                       <jj>
550 if q$=li$ then d=d+1           <gk>
560 if q$=re$ then d=d-1:if d<0 th
en d=0                             <kb>
570 if q$=cr$ then tp=len(w$)-d:en
$="-":printcl$:return              <ga>
580 goto510                        <ap>
590 rem ========================== <ce>
600 rem  p r o g r a m m e n d e   <gm>
610 rem ========================== <id>
```

SCHALTER.C16

```
10 rem schalter ===============c16 <im>
20 rem *************************** <cb>
30 rem ** abfrage der optischen ** <gg>
40 rem ******* schalter ********** <ah>
50 scnclr:for i=1 to 10:char,1,i,"
schalter"+str$(i-1)+":":next       <gb>
60 t0=ti                           <ag>
70 do:a=joy(1):b=joy(2):loop until
 a+b                               <ij>
80 dt=ti-t0                        <kc>
90 if(a+b)=128 then a=-9*(a=128):b
=-9*(b=128)                        <mb>
100 sw=(a+b+1)/2-5*(b>0)           <ga>
110 for i=1 to 10:char,15,i,str$(-
(i=sw)):next                       <fo>
120 goto60                         <ia>
130 rem ========================== <np>
140 rem  p r o g r a m m e n d e   <fe>
150 rem ========================== <nj>
```

fernt, wird in das Unterprogramm ab Zeile 480 verzweigt.
Zeile 480: Die nächsten fünf Zeichen werden aus dem Speicher geholt, damit man beim Trennen die Fortsetzung des Wortes sehen kann.
Zeile 490: Ist das Folgezeichen ein Leerzeichen, so wird die Zeile selbständig abgeschlossen.
Zeile 500–550: Die überlange Zeile wird angezeigt. Der Wort- oder Satzteil, der über die maximale Zeilenlänge hinausgeht, erscheint revers.

ZWEISPALTIG DRUCKEN

SCRIPT/PLUS ist zwar sehr komfortabel und auch das im Plus4 eingebaute Textsystem erlaubt (in gewissen Grenzen) die Erstellung von größeren Texten. Natürlich kann man für diesen Preis kein Desktop-Publishing-Programm verlangen. Aber es können auf einfache Weise mit SCRIPT/PLUS auch zweispaltige Texte erstellt werden. Dazu muß der Text zuerst vollkommen fertig eingegeben sein. Dann begrenzt man die Zeilenlänge zum Beispiel auf 40 Zeichen mit dem Programm **SILBENTRENNUNG**. Im Eingangsmenü von SCRIPT/PLUS wird die gesamte Zeilenlänge (zwei Spalten + Abstand + Rand) eingestellt. Nachdem der Text eingeladen wurde (Wordwrap ausschalten) ist der Text auf die linke Hälfte der verfügbaren Breite beschränkt. Jetzt kommt eine außergewöhnliche Eigenschaft von SCRIPT/PLUS zur Anwendung – nur wenige Textsysteme haben diese. Mit Esc und Shift R kann ein kompletter Textblock markiert werden (es erscheint **SET COLUMN RANGE**) und nach Drücken von Return kann dieser mit den Cursortasten verschoben werden. Angenommen, es sollen 50 Zeilen auf eine Seite gedruckt werden, so muß der Text ab Zeile 51 bis 100 als Block markiert und zuerst nach rechts und dann nach oben verschoben werden, bis die beiden Spalten nebeneinander stehen. Vorsicht! Der markierte Block kann den stehenden Text überschreiben, wenn der Block über diesen Text geschoben wird. Dieser Vorgang wiederholt sich, bis alle Seiten zweispaltig im Speicher stehen. Dabei kann auch Platz für Bilder gelassen werden. Nach Einfügen von Formatzeichen und Seite-Ende-Kennung kann jetzt der zweispaltige Text ausgedruckt werden. □

Sprites und Animation

Hier ist sie, die BASIC-Erweiterung, mit der Sie, sofern Ihr Rechner mit mehr als 16 KByte Hauptspeicher ausgerüstet ist, richtige Sprites erzeugen können. 26 Befehle bringen eine Menge Bewegung ins Spiel.

Jetzt gibt es auch für den C16 Sprites, die den Hintergrund nicht überschreiben und pixelweise positioniert werden können. Außerdem stellen wir jede Menge Befehle vor, die das Programmieren von Animation, Scrolling und vielem mehr ermöglichen.

Vorab noch einige Hinweise:

1. Alle Befehlsworte beginnen mit einer Raute (#).
2. Im Direktmodus und nach den BASIC-Befehlen THEN und ELSE muß zusätzlich ein Doppelpunkt eingegeben werden:
   IFX=0THEN:#SPRITEOFF0
3. Der Befehl GRAPHIC CLR darf nicht mehr benutzt werden, da sonst S&A BASIC zerstört würde.
4. Bevor man die Sprites benutzt, muß man :#ZTRANS120 :#CHN120 : POKE65287,136 eingeben.
5. Aufbau eines Sprites:
   Ein Sprite ist ein aus neun Zeichen zusammengesetzter Block.
   Es werden aber nur zwei mal zwei Zeichen als Sprite dargestellt.
6. Sprites dürfen sich nicht überlagern, da es sonst zu Grafikverfälschungen kommt.

DIE S&A-BASIC-BEFEHLE

1. **#SETSPRITE Nr, XG, YG, XF, YF**
   [Nr=0–7; XG=0–37; YG=0–22; XF, YF=0–8]

Setzt das Sprite Nr auf die Bildschirmposition XG; YG. XF und YF bieten die Möglichkeit, das Sprite nach der groben Positionierung auf die Feinposition XF; YF zu setzen.

2. **#SPRITEOFF NR**
   [NR=0–7]

Schaltet das Sprite Nr ab.

3. **#ANIMATE BC, R**
   [BC=0–255; R=0–3]

Das Zeichen mit dem Bildschirmcode BC wird um ein Pixel in die Richtung R verschoben. Was die einzelnen R-Werte bewirken, ist unter „Richtungswerte" beschrieben.

4. **#SPIEGEL BC, RH**
   [BC=0–255; RH=0–1]

Das Zeichen BC wird horizontal (RH=0) beziehungsweise vertikal (RH=1) gespiegelt.

5. **#FILL XO, YO, XU, YU, Z**
   [XO, XU=0–39; YO,YU=0–24; Z=0–255]

Füllt den Bildschirmausschnitt mit den Koordinaten XO; YO und XU; YU mit den CHR$-Code des Zeichens Z.

6. **#SPALTE Flag1**
   [Flag1=0–1]

Der Bildschirm wird abhängig von Flag1 auf 38 beziehungsweise 40 Spalten gesetzt (siehe auch unter Flag1).

7. **#ZEILE Flag1**
   [Flag1=0–1]

Der Bildschirm wird, abhängig von Flag1, auf 24- oder 25-Zeilen-Format gebracht (siehe auch unter Flag1).

8. **#SCROLL R, XO, YO, XU, YU, Flag2**
   [R=0–3; XO, XU=0–39; YO,YU=0–24; Flag2=0–1]

Scrollt oder rollt einen Bildschirmausschnitt mit den Koordinaten XO; YO und XU; YU in die Richtung R (siehe auch unter „Richtung", Flag2).

9. **#SOFTV Pos**
   [Pos=0–7]

Setzt den Bildschirm auf die vertikale Smooth-Scrolling-Position pos.

10. **#SOFTH Pos**
    [Pos=0–7]

Hat dieselbe Wirkung wie SoftV, nur in horizontaler Richtung.

n mit BASIC für C16/P 4

11. #CHN ADR
 [ADR=0–255]

Schaltet den selbstdefinierten Zeichensatz mit der High-Adresse ADR an (siehe auch ADR).

12. #CHOFF

Schaltet denselben aus und gleichzeitig den Original-Zeichensatz an.

13. #ZTRANS ADR
 [ADR=0–255]

Kopiert den Original-Zeichensatz an die High-Adresse ADR ins RAM (siehe auch ADR).

14. #ZLINK BC, MUSTER
 [BC=0–255; Muster=0–255]

Das Zeichen mit dem Bildschirmcode BC wird mit dem Muster EOR-verknüpft.

15. #MCN

Schaltet den Multicolor-Modus an.

16. #MCOFF

Schaltet ihn aus.

17. #ECMN

Schaltet den erweiterten Hintergrund-Modus (=Extended-Color-Mode) an.

18. #ECMOFF
Schaltet ihn aus.

19. #ZDESIGN BC, x, x, x, x, x, x, x, x
 [BC,x=0–255]
Definiert ein neues Zeichen mit dem Bildschirmcode BC und den acht Byte x.

20. #SDESIGN Nr, Bnr, x, x, x, x, x, x, x, x
 [Nr=0–7; Bnr=0–3; x=0–255]
Definiert den Block Bnr des Sprites Nr durch die acht Byte x.
Bnr=0 linke obere Sprite-Ecke
Bnr=1 linke untere Sprite-Ecke
Bnr=2 rechte obere Sprite-Ecke
Bnr=3 rechte untere Sprite-Ecke

21. #SREVERS Nr, Bnr
 [Nr=0–7; Bnr=0–3]

Reversiert den Block Bnr des Sprites (siehe auch #SDESIGN BNR).

22. #ECMCOL F1, F2, F3
 [F1, F2, F3=0–255]

Legt die drei zusätzlichen Farben für den ECM-MODE fest.

23. #MCCOL F1, F2
 [F1, F2=0–255]

Legt die zwei zusätzlichen Farben des MC-Modes fest.

24. #MELODY Mnr, Tg, L, TH, TL, . . .
 [Mnr=0–4; Tg=0–2; L=0–8; TH=0–1023; TL=0–255]

Definiert die Melodie Mnr auf dem Tongenerator Tg mit der Länge L und den dazugehörigen Noten (TH=Tonhöhe, TL=Tonlänge).

25. #PLAY Mnr
 [Mnr=0–4]

Spielt die Melodie Mnr, die zuvor definiert wurde.

26. #SHOW

Zeigt alle S&A-BASIC-Befehle auf dem Bildschirm an.

RICHTUNGSWERTE

Bei R=0 wird nach links, bei R=1 nach rechts verschoben. Nach oben wird bei R=2 verlagert, bei R=3 folglich nach unten

Flag 1
Flag 1=0 – Der Bildschirm wird verkleindert.
Flag 1=1 – Der Bildschirm hat seine normale Größe.

Flag 2
Flag 2 bestimmt, ob der Bildschirm gerollt:
Flag 2 < 128 oder gescrollt wird:
Flag 2 ≥128

Lesen Sie bitte weiter auf Seite 42

```
S&A-BASIC========================P4
(P) COMMODORE WELT TEAM
===================================
(C) BY MICHAEL INDEN

PLUS4 (C16/116 + 64 KB)
===================================
```

Das Programm ist mit dem Maschinensprachmonitor TEDMON unter Zuhilfenahme des Pruefsummenprogrammes CHECKMON einzugeben und auf Diskette mit S"S&A BASIC",8,1001,2005 abzuspeichern, auf Kassette mit S"S&A BASIC",1,1001,2005.
Danach kann das Programm wie ein BASIC-Programm gehandhabt werden.

```
===================================
>1000 00 23 10 0a 00 8f 20 53 :<46>
>1008 26 41 20 42 41 53 49 43 :<bd>
>1010 3d 3d 3d 3d 3d 3d 3d 3d :<0b>
>1018 3d 3d 3d 3d 3d 3d 3d 3d :<3b>
>1020 50 34 00 44 10 14 00 8f :<f5>
>1028 20 28 50 29 20 42 59 20 :<bb>
>1030 43 4f 4d 4d 4f 44 4f 52 :<eb>
>1038 45 20 57 45 4c 54 20 54 :<fe>
>1040 45 41 4d 00 66 10 1e 00 :<6b>
>1048 8f 20 3d 3d 3d 3d 3d 3d :<73>
>1050 3d 3d 3d 3d 3d 3d 3d 3d :<8c>
>1058 3d 3d 3d 3d 3d 3d 3d 3d :<c0>
>1060 3d 3d 3d 3d 3d 00 81 10 :<a1>
>1068 28 00 8f 20 28 43 29 20 :<69>
>1070 42 59 20 4d 49 43 48 41 :<ef>
>1078 45 4c 20 49 4e 44 45 4e :<66>
>1080 00 87 10 32 00 8f 00 a5 :<b4>
>1088 10 3c 00 8f 20 50 4c 55 :<cf>
>1090 53 34 20 28 43 31 36 2f :<1d>
>1098 31 31 36 20 2b 20 36 34 :<04>
>10a0 20 4b 42 29 00 c7 10 46 :<58>
>10a8 00 8f 20 3d 3d 3d 3d 3d :<12>
>10b0 3d 3d 3d 3d 3d 3d 3d 3d :<d0>
>10b8 3d 3d 3d 3d 3d 3d 3d 3d :<7f>
>10c0 3d 3d 3d 3d 3d 3d 00 e2 :<90>
>10c8 10 50 00 9e 20 34 33 35 :<e6>
>10d0 38 3a 8f 20 4d 41 53 43 :<0c>
>10d8 48 49 4e 45 4e 43 4f 44 :<1e>
>10e0 45 00 04 11 5a 00 8f 20 :<08>
>10e8 3d 3d 3d 3d 3d 3d 3d 3d :<5f>
>10f0 3d 3d 3d 3d 3d 3d 3d 3d :<4f>
>10f8 3d 3d 3d 3d 3d 3d 3d 3d :<00>
>1100 3d 3d 3d 00 00 00 a9 4f :<8d>
>1108 85 2c 85 2e 20 9a 8a c6 :<60>
>1110 75 a9 4c 85 d0 a9 11 85 :<6d>
>1118 d1 a9 b9 85 d2 a9 0e 85 :<8c>
>1120 d3 a9 00 85 d4 a9 40 85 :<60>
>1128 d5 a5 d2 aa a0 00 b1 d0 :<74>
>1130 91 d4 c8 d0 f9 e6 d1 e6 :<bb>
>1138 d5 ca d0 f2 a6 d2 b1 d0 :<f2>
>1140 91 d4 c8 ca d0 f8 8e 00 :<df>
>1148 4f 4c 00 40 a9 fd a2 40 :<7e>
>1150 8d 08 03 8e 09 03 20 67 :<73>
>1158 c5 a2 00 bd 33 40 20 d2 :<dc>
>1160 ff e8 e0 cb d0 f5 a9 00 :<7c>
>1168 8d b9 4e a9 4e 85 2c 85 :<31>
>1170 2e a9 ba 85 2b a9 a6 85 :<f1>
>1178 2d 20 7b 8a 4c 03 87 23 :<47>
>1180 23 23 23 23 23 23 23 23 :<11>
>1188 23 23 23 23 23 23 23 23 :<01>
>1190 23 23 23 23 23 23 23 23 :<32>
>1198 23 23 23 23 23 23 23 23 :<21>
>11a0 23 23 23 23 23 23 23 23 :<52>
>11a8 23 20 53 26 41 20 42 20 :<bc>
>11b0 41 20 53 20 49 20 43 20 :<3a>
>11b8 20 46 4f 52 20 50 4c 55 :<86>
>11c0 53 34 2f 43 31 36 20 28 :<1a>
>11c8 36 34 4b 29 20 23 23 23 :<56>
>11d0 23 20 20 20 20 20 20 20 :<a5>
>11d8 20 20 20 20 20 20 20 20 :<b2>
>11e0 20 20 20 20 20 20 20 20 :<c2>
>11e8 20 20 20 20 20 20 20 20 :<d2>
>11f0 20 20 20 20 20 23 23 23 :<14>
>11f8 23 20 57 52 49 54 54 45 :<07>
>1200 4e 20 42 59 20 12 4d 49 :<66>
>1208 43 48 41 45 4c 20 20 49 :<92>
>1210 4e 44 45 4e 92 20 49 4e :<22>
>1218 20 38 37 2f 38 38 20 23 :<29>
>1220 23 23 23 23 23 23 23 23 :<52>
>1228 23 23 23 23 23 23 23 23 :<42>
>1230 23 23 23 23 23 23 23 23 :<ed>
>1238 23 23 23 23 23 23 23 23 :<62>
>1240 23 23 23 23 23 23 23 23 :<92>
>1248 23 20 73 04 c9 23 f0 06 :<df>
>1250 20 79 04 4c d9 8b 20 73 :<d6>
>1258 04 a2 00 86 19 a2 00 a0 :<d5>
>1260 00 bd 12 4a c9 80 d0 03 :<f8>
>1268 4c f6 49 d1 3b d0 26 e8 :<11>
>1270 c8 bd 12 4a d0 f5 18 98 :<fc>
>1278 65 3b 85 3b 90 03 e6 3c :<be>
>1280 18 a5 19 0a aa bd bf 4a :<50>
>1288 a8 bd c0 4a 8d 48 41 8c :<ee>
>1290 47 41 4c 90 47 e8 bd 12 :<54>
>1298 4a d0 fa e8 e6 19 4c 13 :<52>
>12a0 41 d7 4b 1f 4c 67 4c af :<5c>
>12a8 4c f7 4c 3f 4d 87 4d cf :<18>
>12b0 4d 20 cb 41 20 ee 49 c9 :<ac>
>12b8 26 b0 42 8d b5 41 20 ee :<29>
>12c0 49 c9 17 b0 38 8d b6 41 :<8b>
>12c8 20 ee 49 c9 09 b0 2e 8d :<e3>
>12d0 b7 41 20 ee 49 c9 09 b0 :<a4>
>12d8 24 8d b8 41 20 d9 41 20 :<79>
>12e0 f2 41 20 ff 41 20 4a 42 :<4a>
>12e8 20 78 42 20 0b 43 20 27 :<93>
>12f0 43 20 db 43 20 33 43 20 :<c5>
>12f8 94 43 4c 04 41 4c 0d 4a :<74>
>1300 00 00 00 00 00 20 cb 41 :<6a>
>1308 20 d9 41 20 ff 41 20 d3 :<84>
>1310 42 20 bf 43 4c 04 41 20 :<6e>
```

```
>1318 84 9d e0 08 90 03 4c 0d :<b3>
>1320 4a 8e b4 41 60 a9 4e a2 :<04>
>1328 5f 8e e9 41 85 1a ad b4 :<1b>
>1330 41 0a 0a 18 69 5f 90 03 :<75>
>1338 e6 1a 18 85 19 60 a0 00 :<ee>
>1340 b9 b5 41 91 19 c8 c0 04 :<d4>
>1348 d0 f6 60 a9 4e 8d 13 43 :<1b>
>1350 8d 19 43 8d 1f 43 8d c5 :<e4>
>1358 43 8d cb 43 20 a4 4e ad :<c5>
>1360 b4 41 8d 9f 4e 0a 0a 0a :<d7>
>1368 18 6d 9f 4e 69 17 90 13 :<ac>
>1370 ee 13 43 ee 19 43 ee 1f :<29>
>1378 43 ee c5 43 ee cb 43 20 :<88>
>1380 ab 4e 18 8d 12 43 8d 18 :<d7>
>1388 43 8d 1e 43 8d c4 43 8d :<fb>
>1390 ca 43 20 b2 4e 60 ad b4 :<ac>
>1398 41 0a aa bd 55 41 85 e2 :<a5>
>13a0 bd 56 41 85 e3 ae b4 41 :<78>
>13a8 86 28 a9 00 85 29 85 72 :<54>
>13b0 a9 48 85 71 20 3c 9a 86 :<d6>
>13b8 d0 a9 7c c0 00 f0 02 a9 :<e9>
>13c0 7d 85 d1 60 a2 00 8e a0 :<30>
>13c8 4e a5 e2 48 a2 01 8e 9f :<77>
>13d0 4e 8a 0a aa a5 d1 95 d1 :<9f>
>13d8 ae 9f 4e e8 e0 09 d0 ee :<42>
>13e0 a2 01 8e 9f 4e 18 a5 d0 :<9c>
>13e8 7d c9 42 90 06 a0 01 8c :<ef>
>13f0 a0 4e 18 48 8a 0a aa 68 :<da>
>13f8 95 d0 ad a0 4e f0 08 f6 :<1c>
>1400 d1 a0 00 8c a0 4e 18 ae :<a1>
>1408 9f 4e e8 e0 0a d0 d3 68 :<81>
>1410 85 e2 4c d3 42 00 08 10 :<64>
>1418 18 20 28 30 38 40 00 a0 :<11>
>1420 01 b1 19 0a aa bd 7c 4b :<32>
>1428 85 e5 bd 7b 4b a0 00 71 :<5c>
>1430 19 85 e4 90 03 18 e6 e5 :<c5>
>1438 a5 e5 85 e7 a5 e4 69 28 :<75>
>1440 85 e6 90 03 18 e6 e7 a5 :<26>
>1448 e7 85 e9 a5 e6 69 28 85 :<b5>
>1450 e8 90 03 18 e6 e9 60 a0 :<ec>
>1458 00 a2 00 b1 e4 9d 29 4e :<d8>
>1460 e8 b1 e6 9d 29 4e e8 b1 :<4b>
>1468 e8 9d 29 4e e8 c8 c0 03 :<fe>
>1470 d0 e9 60 a0 00 b1 e2 91 :<ec>
>1478 d0 c8 c0 72 d0 f7 60 a0 :<3a>
>1480 00 a5 d1 8d 80 43 a5 d0 :<f9>
>1488 8d 7f 43 8c 9f 4e 98 0a :<b7>
>1490 aa b5 d1 8d 83 43 8d 80 :<dc>
>1498 43 b5 d0 8d 82 43 8d 7f :<d2>
>14a0 43 8d 7f 43 ac 9f 4e a9 :<50>
>14a8 78 8d 7d 43 b9 29 4e 18 :<12>
>14b0 0a 90 03 20 8f 43 0a 90 :<27>
>14b8 03 20 8f 43 0a 90 03 20 :<80>
>14c0 8f 43 8d 7c 43 a2 00 bd :<04>
>14c8 08 78 1d d0 7c 9d d0 7c :<0f>
>14d0 e8 e0 08 d0 f2 c8 c0 09 :<88>
>14d8 d0 b1 60 ee 7d 43 18 60 :<3a>
>14e0 ae b4 41 8a 0a 0a 0a 6d :<cf>
>14e8 b4 41 69 80 8d a1 4e 69 :<76>
>14f0 03 8d a2 4e ad a1 4e a0 :<4d>
>14f8 00 91 e4 69 01 91 e6 69 :<4b>
>1500 01 91 e8 69 01 c8 c0 03 :<b6>
>1508 d0 ef 60 a0 00 a2 00 bd :<04>
>1510 29 4e 91 e4 e8 bd 29 4e :<03>
>1518 91 e6 e8 bd 29 4e 91 e8 :<83>
>1520 e8 c8 c0 03 d0 e9 60 ae :<bc>
>1528 b7 41 e0 00 f0 0f ea ea :<6e>
>1530 ea a2 00 20 2b 44 e8 8a :<97>
>1538 cd b7 41 d0 f6 ae b8 41 :<eb>
>1540 e0 00 f0 0f ea ea ea a2 :<75>
>1548 00 20 08 44 e8 8a cd b8 :<27>
>1550 41 d0 f6 60 a0 01 b1 d0 :<9f>
>1558 48 b1 d6 48 b1 dc 88 91 :<f3>
>1560 dc 68 91 d6 68 91 d0 c8 :<f6>
>1568 c8 c0 18 d0 e9 88 a9 00 :<c7>
>1570 91 d0 91 d6 91 dc 60 a0 :<87>
>1578 00 b1 e0 4a 91 e0 b1 de :<f4>
>1580 4a 91 de b1 dc 4a 91 dc :<e9>
>1588 18 b1 da 4a 91 da 90 07 :<08>
>1590 a9 7f 71 e0 91 e0 18 b1 :<16>
>1598 d8 4a 91 d8 90 07 a9 7f :<4b>
>15a0 71 de 91 de 18 b1 d6 4a :<ee>
>15a8 91 d6 90 07 a9 7f 71 dc :<ce>
>15b0 91 dc 18 b1 d4 4a 91 d4 :<39>
>15b8 90 07 a9 7f 71 da 91 da :<a7>
>15c0 18 b1 d2 4a 91 d2 90 07 :<94>
>15c8 a9 7f 71 d8 91 d8 18 b1 :<83>
>15d0 d0 4a 91 d0 90 07 a9 7f :<b3>
>15d8 71 d6 91 d6 18 c8 c0 08 :<87>
>15e0 d0 97 60 20 a4 49 20 bc :<81>
>15e8 49 20 ee 49 e0 00 f0 0f :<f3>
>15f0 e0 01 f0 1c e0 02 f0 2b :<c8>
>15f8 e0 03 f0 41 4c 0d 4a a0 :<f7>
>1600 00 b1 19 0a 69 00 91 19 :<75>
>1608 c8 c0 08 d0 f4 4c 04 41 :<7e>
>1610 a0 00 b1 19 4a 90 02 69 :<3d>
>1618 7f 91 19 c8 c0 08 d0 f2 :<7c>
>1620 4c 04 41 a0 00 b1 19 48 :<08>
>1628 a0 01 b1 19 88 91 19 c8 :<58>
>1630 c8 c0 08 d0 f5 68 a0 07 :<24>
>1638 91 19 4c 04 41 a0 07 b1 :<5d>
>1640 19 48 a0 06 b1 19 c8 91 :<e7>
>1648 19 88 88 c0 ff d0 f5 68 :<4c>
>1650 a0 00 91 19 4c 04 41 20 :<0b>
>1658 a4 49 20 bc 49 20 ee 49 :<12>
>1660 e0 02 90 03 4c 0d 4a e0 :<fa>
>1668 01 f0 29 a5 19 8d 34 45 :<eb>
>1670 8d 3a 45 a5 1a 8d 35 45 :<fb>
>1678 8d 3b 45 a2 00 a0 07 bd :<e4>
>1680 00 0c 48 b1 19 9d 00 0c :<05>
>1688 68 91 19 88 e8 e0 04 d0 :<bc>
>1690 ee 4c 04 41 a0 00 b1 19 :<da>
>1698 29 0f 20 6d 45 0a 0a 0a :<93>
>16a0 0a 85 1f b1 19 4a 4a 4a :<c6>
>16a8 4a 20 6d 45 18 65 1f 91 :<eb>
>16b0 19 c8 c0 08 d0 e0 4c 04 :<2d>
```

```
>16b8 41 a2 00 dd 7d 45 f0 05 :<e1>
>16c0 e8 e0 10 d0 f6 bd 8d 45 :<ba>
>16c8 60 00 01 02 03 04 05 06 :<10>
>16d0 07 08 09 0a 0b 0c 0d 0e :<7f>
>16d8 0f 00 08 04 0c 02 0a 06 :<9d>
>16e0 0e 01 09 05 0d 03 0b 07 :<ab>
>16e8 0f 20 84 9d 8a cd f2 45 :<15>
>16f0 90 03 4c 0d 4a 8d f7 45 :<3d>
>16f8 a0 01 98 48 20 ee 49 68 :<4f>
>1700 a8 8a d9 f2 45 90 03 4c :<e2>
>1708 0d 4a 99 f7 45 c8 c0 05 :<a2>
>1710 d0 e8 ae f8 45 ac f7 45 :<6b>
>1718 18 20 f0 ff ae f7 45 86 :<25>
>1720 21 ad fb 45 20 d2 ff a5 :<36>
>1728 21 e6 21 cd f9 45 90 f1 :<d3>
>1730 ad f8 45 ee f8 45 cd fa :<fd>
>1738 45 90 d7 4c 04 41 28 19 :<82>
>1740 28 19 c0 00 0b 1d 0a 41 :<68>
>1748 20 84 9d 8a f0 07 c9 01 :<28>
>1750 f0 0b 4c 0d 4a ad 07 ff :<be>
>1758 29 f7 4c 16 46 ad 07 ff :<07>
>1760 09 08 8d 07 ff 4c 04 41 :<56>
>1768 20 84 9d 8a f0 07 c9 01 :<68>
>1770 f0 0b 4c 0d 4a ad 06 ff :<be>
>1778 29 f7 4c 36 46 ad 06 ff :<07>
>1780 09 08 8d 06 ff 4c 04 41 :<8e>
>1788 20 84 9d 8a 85 20 20 ee :<63>
>1790 49 c9 28 b0 2a 85 1b 20 :<4f>
>1798 ee 49 c9 19 b0 21 85 1c :<43>
>17a0 20 ee 49 c9 28 b0 18 85 :<a4>
>17a8 1d 20 ee 49 c9 19 b0 0f :<ea>
>17b0 85 1e 20 ee 49 85 21 e6 :<8a>
>17b8 1e a5 20 c9 04 90 03 4c :<d5>
>17c0 0d 4a c9 03 d0 05 c6 1e :<91>
>17c8 4c 2a 47 c9 02 d0 05 c6 :<ea>
>17d0 1e 4c ef 46 c9 00 f0 05 :<66>
>17d8 e6 1d 4c 93 46 c6 1b a4 :<80>
>17e0 1c 20 e0 49 a5 20 f0 06 :<46>
>17e8 20 b1 46 4c a5 46 20 d0 :<d8>
>17f0 46 a4 1c c8 84 1c c4 1e :<b0>
>17f8 90 e5 4c 04 41 c6 1d a4 :<39>
>1800 1d e6 1d b1 19 aa 24 21 :<63>
>1808 10 02 a2 20 a4 1b b1 19 :<2a>
>1810 48 8a 91 19 68 aa c8 c4 :<2b>
>1818 1d d0 f3 60 e6 1b a4 1b :<2c>
>1820 c6 1b b1 19 aa 24 21 10 :<4b>
>1828 02 a2 20 a4 1d b1 19 48 :<25>
>1830 8a 91 19 68 aa 88 c4 1b :<95>
>1838 d0 f3 60 a4 1c 20 e0 49 :<3f>
>1840 e6 1d a4 1b b1 19 99 af :<4c>
>1848 4b c8 c4 1d d0 f6 20 c7 :<ae>
>1850 49 a9 de 8d 0c 47 20 f6 :<bb>
>1858 00 24 21 30 13 a4 1e 20 :<70>
>1860 e0 49 a4 1b a4 1b b9 af :<c9>
>1868 4b 91 19 c8 c4 1d d0 f6 :<c8>
>1870 20 70 de 4c 04 41 a4 1e :<f3>
>1878 20 e0 49 e6 1d a4 1b b1 :<42>
>1880 19 99 af 4b c8 c4 1d d0 :<ae>
>1888 f6 20 c7 49 20 04 df 24 :<11>
>1890 21 30 11 a4 1c 20 e0 49 :<7e>
>1898 a4 1b b9 af 4b 91 19 c8 :<74>
>18a0 c4 1d d0 f6 20 70 de 4c :<d2>
>18a8 04 41 20 84 9d e0 08 90 :<22>
>18b0 03 4c 0d 4a 86 19 ad 06 :<43>
>18b8 ff 29 f8 05 19 8d 06 ff :<25>
>18c0 4c 04 41 20 84 9d e0 08 :<bb>
>18c8 90 03 4c 0d 4a 86 19 ad :<eb>
>18d0 07 ff 29 f8 05 19 8d 07 :<39>
>18d8 ff 4c 04 41 20 84 9d 86 :<10>
>18e0 21 8e 13 ff a9 c0 8d 12 :<29>
>18e8 ff c6 21 a5 21 85 38 a9 :<ff>
>18f0 ff 85 37 4c 04 41 a9 d0 :<bb>
>18f8 8d 13 ff a9 c4 8d 12 ff :<ef>
>1900 4c 04 41 20 84 9d 86 1a :<cd>
>1908 a2 d0 86 1c a2 00 86 19 :<6b>
>1910 86 1b a2 08 a0 00 b1 1b :<c0>
>1918 91 19 88 c0 00 d0 f7 e6 :<d3>
>1920 1a e6 1c ca e0 00 d0 ec :<eb>
>1928 4c 04 41 20 a4 49 20 bc :<08>
>1930 49 20 ee 49 8d f0 47 a0 :<6f>
>1938 00 b1 19 49 ff 91 19 c8 :<3e>
>1940 c0 08 d0 f5 4c 04 41 ad :<7c>
>1948 07 ff 09 10 8d 07 ff 4c :<f2>
>1950 04 41 ad 07 ff 29 ef 8d :<64>
>1958 07 ff 4c 04 41 ad 06 ff :<ca>
>1960 09 40 8d 06 ff 4c 04 41 :<c4>
>1968 ad 06 ff 29 bf 8d 06 ff :<86>
>1970 4c 04 41 20 a4 49 20 bc :<7d>
>1978 49 a2 00 86 21 20 ee 49 :<f7>
>1980 a4 21 91 19 a6 21 e8 e0 :<73>
>1988 08 d0 f0 4c 04 41 20 48 :<80>
>1990 48 4c 75 48 20 84 9d e0 :<ba>
>1998 08 90 03 4c 0d 4a 8e b4 :<39>
>19a0 41 20 4a 42 20 78 42 20 :<a8>
>19a8 ee 49 aa e0 04 90 03 4c :<29>
>19b0 0d 4a a5 e2 18 7d 8a 48 :<f8>
>19b8 85 19 a9 00 65 e3 85 1a :<05>
>19c0 60 a0 00 8c 9f 4e 20 ee :<38>
>19c8 49 ac 9f 4e 91 19 c8 c0 :<76>
>19d0 08 d0 f0 4c 04 41 08 10 :<f6>
>19d8 20 28 20 48 48 a0 00 b1 :<ee>
>19e0 19 49 ff 91 19 c8 c0 08 :<8b>
>19e8 d0 f5 4c 04 41 20 84 9d :<2d>
>19f0 8e 16 ff 20 ee 49 8d 17 :<28>
>19f8 ff 20 ee 49 8d 18 ff 4c :<0b>
>1a00 04 41 20 84 9d 8e 17 ff :<14>
>1a08 20 ee 49 8d 18 ff 4c 04 :<1f>
>1a10 41 20 84 9d 8a c9 05 90 :<89>
>1a18 03 4c 0d 4a 0a 85 1b 0a :<5f>
>1a20 0a 85 1d 0a 65 1b 65 1d :<95>
>1a28 69 f3 85 19 a9 4a 85 1a :<8a>
>1a30 20 ee 49 c9 03 90 03 4c :<8a>
>1a38 0d 4a a0 00 91 19 20 ee :<d5>
>1a40 49 c9 09 90 03 4c 0d 4a :<f7>
>1a48 85 21 0a 69 02 65 21 85 :<9b>
>1a50 21 a0 01 91 19 c8 84 20 :<5f>
```

```
>1a58 20 91 94 20 e1 9d a5 15 :<f4>
>1a60 c9 04 90 03 4c 0d 4a a4 :<b0>
>1a68 20 a5 14 91 19 c8 a5 15 :<71>
>1a70 91 19 c8 84 20 20 ee 49 :<99>
>1a78 a4 20 91 19 c8 c4 21 d0 :<ab>
>1a80 d5 4c 04 41 20 84 9d 8a :<a1>
>1a88 c9 05 90 03 4c 0d 4a 0a :<16>
>1a90 85 1b 0a 0a 85 1d 0a 65 :<a0>
>1a98 1b 65 1d 69 f3 85 19 a9 :<61>
>1aa0 4a 85 1a a0 01 b1 19 85 :<ea>
>1aa8 21 a2 08 20 c0 b8 a0 02 :<c6>
>1ab0 84 20 a0 00 b1 19 85 80 :<cd>
>1ab8 a4 20 b1 19 85 7e c8 b1 :<cc>
>1ac0 19 85 7f c8 84 20 b1 19 :<f4>
>1ac8 a8 a9 00 20 61 b8 a4 20 :<bc>
>1ad0 c8 c4 21 d0 db 4c 04 41 :<c6>
>1ad8 a2 00 a0 00 bd 12 4a 30 :<a9>
>1ae0 0c d0 03 a9 20 c8 20 d2 :<c4>
>1ae8 ff e8 4c 90 49 4c 04 41 :<ab>
>1af0 20 84 9d 86 28 a2 08 86 :<1c>
>1af8 71 a2 00 86 29 86 72 18 :<d7>
>1b00 20 3c 9a 84 1a 86 19 60 :<02>
>1b08 ad 13 ff 18 e9 01 65 1a :<8f>
>1b10 85 1a 60 a5 1b 8d e7 07 :<1b>
>1b18 c6 1d a5 1d 8d e8 07 e6 :<41>
>1b20 1d a5 1c 8d e6 07 a5 1e :<15>
>1b28 8d e5 07 60 98 0a a8 b9 :<32>
>1b30 7b 4b 85 19 b9 7c 4b 85 :<5b>
>1b38 1a 60 20 91 94 20 84 9d :<9e>
>1b40 8a 60 a9 01 85 24 a9 4a :<ed>
>1b48 85 25 4c da 86 4e 4f 20 :<40>
>1b50 53 26 41 20 42 41 53 49 :<16>
>1b58 c3 a2 0e 4c 86 86 53 45 :<dd>
>1b60 54 53 50 52 49 54 45 00 :<16>
>1b68 53 50 52 49 54 45 4f 46 :<c5>
>1b70 46 00 41 4e 49 4d 41 54 :<e1>
>1b78 45 00 53 50 49 45 47 45 :<ad>
>1b80 4c 00 46 49 4c 4c 00 53 :<b3>
>1b88 50 41 4c 54 45 00 5a 45 :<5e>
>1b90 49 4c 45 00 53 43 52 4f :<ec>
>1b98 4c 4c 00 53 4f 46 54 56 :<89>
>1ba0 00 53 4f 46 54 48 00 43 :<40>
>1ba8 48 4e 00 43 48 4f 46 46 :<05>
>1bb0 00 5a 54 52 41 4e 53 00 :<7f>
>1bb8 5a 4c 49 4e 4b 00 4d 43 :<76>
>1bc0 4e 00 4d 43 4f 46 46 00 :<89>
>1bc8 45 43 4d 4e 00 45 43 4d :<3b>
>1bd0 4f 46 46 00 5a 44 45 53 :<ba>
>1bd8 49 47 4e 00 53 44 45 53 :<56>
>1be0 49 47 4e 00 53 52 45 56 :<48>
>1be8 45 52 53 00 45 43 4d 43 :<53>
>1bf0 4f 4c 00 4d 43 43 4f 4c :<53>
>1bf8 00 4d 45 4c 4f 44 59 00 :<e9>
>1c00 50 4c 41 59 00 53 48 4f :<d6>
>1c08 57 00 80 65 41 b9 41 97 :<c3>
>1c10 44 0b 45 9d 45 fc 45 1c :<c3>
>1c18 46 3c 46 5e 47 77 47 90 :<13>
>1c20 47 aa 47 b7 47 df 47 fb :<69>
>1c28 47 06 48 11 48 1c 48 27 :<d6>
>1c30 48 42 48 8e 48 a1 48 b6 :<72>
>1c38 48 c5 48 38 49 8c 49 00 :<99>
>1c40 00 00 00 00 00 00 00 00 :<9c>
>1c48 00 00 00 00 00 00 00 00 :<ac>
>1c50 00 00 00 00 00 00 00 00 :<bc>
>1c58 00 00 00 00 00 00 00 00 :<cc>
>1c60 00 00 00 00 00 00 00 00 :<dc>
>1c68 00 00 00 00 00 00 00 00 :<ec>
>1c70 00 00 00 00 00 00 00 00 :<fc>
>1c78 00 00 00 00 00 00 00 00 :<0c>
>1c80 00 00 00 00 00 00 00 00 :<1d>
>1c88 00 00 00 00 00 00 00 00 :<2d>
>1c90 00 00 00 00 00 00 00 00 :<3d>
>1c98 00 00 00 00 00 00 00 00 :<4d>
>1ca0 00 00 00 00 00 00 00 00 :<5d>
>1ca8 00 00 00 00 00 00 00 00 :<6d>
>1cb0 00 00 00 00 00 00 00 00 :<7d>
>1cb8 00 00 00 00 00 00 00 00 :<8d>
>1cc0 00 00 00 00 00 00 00 00 :<9d>
>1cc8 0c 28 0c 50 0c 78 0c a0 :<8f>
>1cd0 0c c8 0c f0 0c 18 0d 40 :<30>
>1cd8 0d 68 0d 90 0d b8 0d e0 :<3d>
>1ce0 0d 08 0e 30 0e 58 0e 80 :<e3>
>1ce8 0e a8 0e d0 0e f8 0e 20 :<3d>
>1cf0 0f 48 0f 70 0f 98 0f c0 :<fc>
>1cf8 0f e8 0f 00 00 00 00 00 :<28>
>1d00 00 00 00 00 00 00 00 00 :<1d>
>1d08 00 00 00 00 00 00 00 00 :<2d>
>1d10 00 00 00 00 00 00 00 00 :<3d>
>1d18 00 00 00 00 00 00 00 00 :<4d>
>1d20 00 00 00 00 00 00 00 00 :<5d>
>1d28 00 00 00 ff ff ff ff ff :<74>
>1d30 ff ff ff ff ff ff ff ff :<7d>
>1d38 ff ff ff 00 00 00 00 00 :<86>
>1d40 00 00 00 ff ff ff ff ff :<a4>
>1d48 ff ff ff ff ff ff ff ff :<ad>
>1d50 ff ff ff 00 00 00 00 00 :<b6>
>1d58 00 00 00 00 00 00 00 00 :<cd>
>1d60 00 00 00 00 00 00 00 00 :<dd>
>1d68 00 00 00 00 00 00 00 00 :<ed>
>1d70 00 00 00 08 1c 1c 1c 3e :<29>
>1d78 3e be ff ff be 3e 3e 1c :<0d>
>1d80 1c 1c 08 00 00 00 00 00 :<92>
>1d88 00 00 00 00 00 00 00 04 :<30>
>1d90 0e 3e ff ff 3e 0e 04 00 :<43>
>1d98 00 00 00 00 00 00 00 00 :<4e>
>1da0 00 00 00 00 00 00 00 00 :<5e>
>1da8 00 00 00 00 00 00 00 00 :<6e>
>1db0 00 00 00 00 00 00 00 00 :<7e>
>1db8 00 00 00 01 02 03 04 05 :<9a>
>1dc0 06 07 08 00 00 00 00 00 :<d2>
>1dc8 00 00 00 00 00 00 00 00 :<ae>
>1dd0 00 00 00 00 00 00 00 00 :<be>
>1dd8 00 00 00 00 00 00 00 00 :<ce>
>1de0 00 00 00 00 00 00 00 00 :<de>
>1de8 00 00 00 00 00 00 00 00 :<ee>
>1df0 00 00 00 00 00 00 00 00 :<fe>
```

```
>1df8 00 00 00 00 00 00 00 00 :<0e>
>1e00 00 00 00 00 00 00 00 00 :<1e>
>1e08 00 00 00 00 00 00 00 00 :<2e>
>1e10 00 00 00 00 00 00 00 00 :<3e>
>1e18 00 00 00 00 00 00 00 00 :<4e>
>1e20 00 00 00 00 00 00 00 00 :<5e>
>1e28 00 00 00 00 00 00 00 00 :<6e>
>1e30 00 00 00 00 00 00 00 00 :<7e>
>1e38 00 00 00 00 00 00 00 00 :<8e>
>1e40 00 00 00 00 00 00 00 00 :<9e>
>1e48 00 00 00 00 00 00 00 00 :<ae>
>1e50 00 00 00 00 00 00 00 00 :<be>
>1e58 00 00 00 00 00 00 00 00 :<ce>
>1e60 00 00 00 00 00 00 00 00 :<de>
>1e68 00 00 00 00 00 00 00 00 :<ee>
>1e70 00 00 00 00 00 00 00 00 :<fe>
>1e78 00 00 00 00 00 00 00 00 :<0e>
>1e80 00 00 00 00 00 00 00 00 :<1f>
>1e88 00 00 00 00 00 00 00 00 :<2f>
>1e90 00 00 00 00 00 00 00 00 :<3f>
>1e98 00 00 00 00 00 00 00 00 :<4f>
>1ea0 00 00 00 00 00 00 00 00 :<5f>
>1ea8 00 00 00 00 00 00 00 00 :<6f>
>1eb0 00 00 00 00 00 00 00 00 :<7f>
>1eb8 00 00 00 00 00 00 00 00 :<8f>
>1ec0 00 00 00 00 00 00 00 00 :<9f>
>1ec8 00 00 00 00 00 00 00 00 :<af>
>1ed0 00 00 00 00 00 00 00 00 :<bf>
>1ed8 00 00 00 00 00 00 00 00 :<cf>
>1ee0 00 00 00 00 00 00 00 00 :<df>
>1ee8 00 00 00 00 00 00 00 00 :<ef>
>1ef0 00 00 00 00 00 00 00 00 :<ff>
>1ef8 00 00 00 00 00 00 00 00 :<0f>
>1f00 00 00 00 00 00 00 00 00 :<1f>
>1f08 00 00 00 00 00 00 00 00 :<2f>
>1f10 00 00 00 00 00 00 00 00 :<3f>
>1f18 00 00 00 00 00 00 00 00 :<4f>
>1f20 00 00 00 00 00 00 00 00 :<5f>
>1f28 00 00 00 00 00 00 00 00 :<6f>
>1f30 00 00 00 00 00 00 00 00 :<7f>
>1f38 00 00 00 00 00 00 00 00 :<8f>
>1f40 00 00 00 00 00 00 00 00 :<9f>
>1f48 00 00 00 00 00 00 00 00 :<af>
>1f50 00 00 00 00 00 00 00 00 :<bf>
>1f58 00 00 00 00 00 00 00 00 :<cf>
>1f60 00 00 00 89 8a 8b 8c 8d :<5f>
>1f68 8e 8f 90 91 01 01 01 01 :<5a>
>1f70 01 01 01 01 01 01 01 01 :<7f>
>1f78 01 01 01 01 01 01 00 00 :<4e>
>1f80 00 00 00 00 00 00 00 00 :<20>
>1f88 00 00 00 00 00 00 00 00 :<30>
>1f90 00 00 00 00 00 00 00 00 :<40>
>1f98 00 00 00 00 00 00 00 00 :<50>
>1fa0 00 00 00 00 00 00 00 00 :<60>
>1fa8 00 00 00 05 05 03 03 00 :<0a>
>1fb0 05 00 00 00 05 00 00 00 :<d5>
>1fb8 00 00 00 00 00 00 00 00 :<90>
>1fc0 00 00 00 00 00 00 00 00 :<a0>
>1fc8 00 00 00 00 00 00 00 00 :<b0>
>1fd0 00 00 00 00 00 00 00 00 :<c0>
>1fd8 00 00 00 00 00 00 00 00 :<d0>
>1fe0 00 00 00 00 00 00 00 00 :<e0>
>1fe8 00 00 00 07 00 92 95 00 :<80>
>1ff0 8d d1 43 8d 62 43 60 ee :<7d>
>1ff8 d1 43 ee 62 43 60 8d d0 :<de>
>2000 43 8d 61 43 60 ff ff ff :<2e>
==================================
P R O G R A M M E N D E
==================================
```

SCA-BASIC

Fortsetzung von Seite 37

ADR

ADR kann zwar theoretisch Werte von 0 bis 255 annehmen. Damit der Rechner nicht abstürzt, sollten aber nur Werte zwischen 16 bis 56 und 79 bis 120 verwendet werden.

Verschiebung bei x-Feinposition: 0 1 2 3 4 5 6 7 8

Verschiebung bei y-Feinposition: 8 7 6 5 4 3 2 1 0

keine Blocknummer — keine Blocknummer — keine Blocknummer

Blocknummer 1 — Blocknummer 3 — keine Blocknummer

Blocknummer 2 — Blocknummer 4 — keine Blocknummer

Die BASIC-Erweiterung geben Sie ein, indem Sie das Programm CHECKMON laden und starten. Anschließend gehen Sie mit MONITOR in den Maschinenmonitor Ihres Rechners. Tippen Sie nur die Zeilen ab, die am Anfang das Zeichen „>" enthalten. Was rechts ab dem Doppelpunkt erscheint, brauchen Sie nicht einzugeben. Die erscheinenden zwei Buchstaben zwischen dem Größer- und Kleinerzeichen dienen zur Kontrolle Ihrer Eingabe. Zum Abspeichern richten Sie sich nach den Anweisungen im Programmvorspann. Nach dem Starten mit RUN kopiert sich das Maschinenprogramm in den Bereich von $4000 bis $4EB8. Die Speicheraufteilung ist die, die sich nach einem Aufruf der hochauflösenden Grafik ergibt. Der BASIC-Anfang wurde vorverlegt auf $4EBA. Nach dem Starten von S&A-BASIC können Sie sich von den neuen Möglichkeiten ein Bild machen, indem Sie das Programm S&A-DEMO ausprobieren, das einige der neuen BASIC-Befehle nützt. □

ZEICHENCREATOR

Eigene Zeichen entwickeln auf dem C16

Mit dem Joystick und menüunterstützt entfällt die lästige Rechnerei. Zeichen editieren wird komfortabel. Besonders wenn Drehungen, Spiegelungen und weitere Funktionen zur Verfügung stehen.

Mit dem Zeichencreator lassen sich sowohl einfarbige als auch Multicolorzeichen komfortabel erstellen und editieren. Sie können Zeichen invertieren, in andere Zeichen kopieren, scrollen, horizontal und vertikal spiegeln, drehen und vieles mehr.
Der Cursor wird mit den Cursortasten und mit dem Joystick in Port 1 beziehungsweise 2 (wählbar, siehe Menüpunkt 1) gesteuert.
Den Zeicheneditor rufen Sie durch Aktivieren des zweiten Menüpunktes auf. Alle Funktionen und Tasten werden rechts auf dem Bildschirm angezeigt.
Den Zeichenmodus können Sie durch Drücken der SHIFT- und „±"-Taste verändern:
Modus 0 = Löschen;
Modus 1 bis 3 = Punktfarbe 1 bis 3.
Um ein Zeichen in ein anderes zu kopieren, drücken Sie c (1), suchen sich mit „±" ein Zeichen aus (2) und drücken Sie Return (3). Editieren Sie das Zeichen (4), drücken Sie erneut c (5) und suchen Sie das Zeichen mit „±" aus, in das das andere Zeichen kopiert werden soll (6). Jetzt drücken Sie „F" (7).
Falls Sie das Zeichen schon editiert haben, führen Sie nur noch Schritt 5 bis 7 aus.
Wenn Sie Ihre Zeichen in eigenen Programmen benutzen wollen, so wählen Sie den Menüpunkt „Zeichendata". Er sollte aber erst dann angewählt werden, nachdem alle Zeichen editiert sind.
Ab Zeile 8000 entstehen Zeichendata mit Einleseprogramm. Jetzt geben Sie Delete –7000 ein, um nur noch das Einleseprogramm und die Data im Speicher zu haben.
Alles weitere wird im Programm erklärt. Vergessen Sie nicht, vor dem ersten RUN abzuspeichern. □

```
10 rem zeichencreator==========c16 <gj>
20 rem (p) commodore welt team     <ho>
30 rem ===========================  <ng>
40 rem (c) by michael inden        <gh>
50 rem                             <pd>
60 rem                             <ah>
70 rem basic v3.5                  <nl>
80 rem c16/116/plus4               <ki>
90 rem ===========================  <jg>
100 poke55,0:poke56,56:clr         <bm>
110 rem mit 64 kb zeile 100 ersetz
en durch 'graphic1,1:clr:graphic0' <ka>
120 rem farben                     <am>
130 :                              <dp>
140 color1,1:color0,2:color4,7,5   <cn>
150 fort=1375to1382:poket,0:next   <nk>
160 :                              <hl>
170 rem zeichenkopierroutine       <ga>
180 :                              <kd>
190 fort=0to32:readp:poke1383+t,p:
next                               <am>
200 data169,208,133,209,169,0,133,
208,133,210,169,56,133,211,162,8,1
60,0,177                           <bj>
210 data208,145,210,136,208,249,23
0,209,230,211,202,208,240,96       <gn>
220 sys1383:gosub3340              <fi>
230 :                              <ah>
240 rem variablen                  <mc>
250 :                              <cp>
260 mf=0:mo=1:po=1:g=1:zd=8060:b=1 <la>
270 p(0)=46:p(1)=81:p(2)=87:p(3)=4
2                                  <fc>
280 dimz%(127),d(63)               <fb>
290 mf$(0)=rn$+"normal"+rf$:mf$(1)
=rn$+"multi"+rf$+" "               <hi>
300 :                              <je>
310 poke65287,peek(65287)and239    <be>
320 :                              <lm>
330 rem zeichensatzeditor          <nn>
340 :                              <oe>
350 rem menue                      <mh>
360 :                              <am>
370 printcl$c4$b6$"***> zeichencre
ator c-16 <***"b6$                 <ig>
380 printc4$b$" ==================
="                                 <fe>
390 printc4$c4$                    <im>
400 printb9$"steuerung...........
1"c4$                              <ci>
410 printb9$"zeichensatzeditor....
2"c4$                              <je>
420 printb9$"zeichendatas.........
3"c4$                              <gk>
430 printb9$"normal/multicolor....
4"c4$                              <ii>
440 printb9$"ende.................
5"                                 <dh>
450 :                              <mb>
460 printc4$c4$b9$"bitte waehlen s
ie [1-5]"                          <lk>
470 a$="":getkeya$                 <fm>
480 a=val(a$)                      <fj>
490 ifa<1ora>5thenprintleft$(qu$,4
):goto460                          <kd>
500 :                              <cf>
510 onagoto540,650,2910,3130,3280  <md>
520 :                              <eo>
530 :                              <gc>
540 rem steuerung                  <km>
550 scnclr                         <di>
560 printc4$c4$c4$" cursosteuerung
 : normal"                         <dj>
570 printc4$c4$" joysticksteuerung
 : port"po" [+]"                   <oh>
580 printc4$c4$" m = "rn$"back to
menue"rf$                          <lf>
590 geta$                          <kl>
600 ifa$="+"thenpo=po+1:ifpo>2then
po=1                               <pp>
610 ifa$="m"thengoto370            <ek>
620 char1,25,6,str$(po)            <ao>
630 goto590                        <dp>
640 :                              <dp>
650 rem zeichensatzeditor          <lm>
660 :                              <gh>
670 printcl$bk$;                   <ki>
680 fort=0to7:print"........B"t:ne
xt                                 <ip>
690 print"CCCCCCCC"zs$             <mc>
700 printc4$"76543210"             <ep>
710 printc4$" code:"using"###";zc:
poke3562,zc                        <pe>
720 print" mode: "mo               <bf>
730 char1,15,0,"funktionstasten:"  <io>
740 char1,15,1,"================"  <hh>
750 char1,15,2,"code = "+re$+"c"+b
k$                                 <bo>
760 char1,15,3,"invert = "+re$+"i"
+bk$                               <ce>
770 char1,15,4,"back to menue = "+
re$+"esc"+bk$                      <ap>
780 char1,15,5,"vertikalspiegeln =
 "+re$+"v"+bk$                     <cm>
790 char1,15,6,"hori.-spiegeln = "
+re$+"h"+bk$                       <bp>
800 char1,15,7,"breite 1/2 = "+re$
+"b"+bk$                           <pp>
810 char1,15,8,"punkt s/l = "+re$+
"space/fire"+bk$                   <bk>
820 char1,15,9,"scroll = "+re$+"s"
+bk$                               <fg>
830 char1,15,10,"home = "+re$+"hom
e"+bk$                             <km>
840 char1,15,11,"clear = "+re$+"cl
ear"+bk$                           <kp>
```

```
850 char1,15,12,"drehung = "+re$+"
u"+bk$                                  <me>
860 char1,15,13,"alt. zeichen zuru
eck = "+re$+"n"+bk$                     <fa>
870 char1,15,14,"zeichen uebernehm
en = "+re$+"f"+bk$                      <go>
880 char1,0,15,"CCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCC"                 <jh>
890 char1,0,18,"scroll = s + curso
r up/down/right/left"                   <od>
900 char1,0,20,"drehung 90/180/270
 = d + 1/2/3"                           <cb>
910 char1,0,22,"return zur ausfueh
rung von drehung und"+b2$               <be>
920 char1,0,23,"scroll druecken"        <oi>
930 gosub2540                           <ik>
940 :                                   <jj>
950 rem bewegung                        <bg>
960 geta$:j=joy(po)                     <ga>
970 poke65299,56:poke65298,0            <gm>
980 ifa$=c3$orj=3thenx=x+b:ifx>8-b
thenx=0                                 <co>
990 ifa$=c1$orj=7thenx=x-b:ifx<0th
enx=8-b                                 <gi>
1000 ifa$=c2$orj=1theny=y-1:ify<0t
heny=7                                  <lj>
1010 ifa$=c4$orj=5theny=y+1:ify>7t
heny=0                                  <ef>
1020 ifa$=he$thenx=0:y=0                <jh>
1030 ifa$=cl$thenx=0:y=0:printhe$;
:fort=0to7:print"........":next         <ke>
1040 ifa$="h"thengosub1370              <hi>
1050 ifa$="v"thengosub1450              <oo>
1060 ifa$="i"thengosub1290              <dh>
1070 ifa$="s"thengosub1530              <lg>
1080 ifa$="u"thengosub1980              <ap>
1090 ifmf=1then1100:elseifa$="b"th
enb=b+1:ifb>2thenb=1                    <nl>
1100 ifa$="c"thengosub2400              <jn>
1110 ifa$="n"thengosub2680              <ej>
1120 ifa$="f"thengosub2800              <ij>
1130 ifa$=ym$thenmo=mo+1:ifmo>gthe
nmo=0                                   <ad>
1140 ifa$=yo$thenmo=mo-1:ifmo<0the
nmo=g                                   <af>
1150 ifa$=chr$(27)thengoto370           <pl>
1160 ifa$=" "orj>=127thengoto1230       <me>
1170 poke3072+y*40+x,peek(3072+y*4
0+x)or128                               <mc>
1180 fort=1to30:next:poke3072+y*40
+x,peek(3072+y*40+x)and127              <ch>
1190 char1,0,16,"breite:":printstr
$(b):poke3562,zc                        <kb>
1200 printhe$left$(qd$,12)" code:"
using"###";zc                           <oo>
1210 print" mode: "mo:char1,1,14,"
<shft.[+/-]>"                           <dj>
1220 goto960                            <fa>
1230 p1=3072+y*40+x:p2=3072+y*40+x
+1                                      <kj>
1240 ifb=2andx=7thengoto960             <oh>
1250 ifb=1thenpokep1,p(mo):goto960      <il>
1260 pokep1,p(mo):pokep2,p(mo)          <co>
1270 goto960                            <hb>
1280 :                                  <ee>
1290 rem invert                         <dj>
1300 fors=0to7                          <hi>
1310 forz=0to7                          <gl>
1320 ifpeek(3072+s*40+z)=46thenpok
e3072+s*40+z,81:goto1340                <hj>
1330 ifpeek(3072+s*40+z)=81thenpok
e3072+s*40+z,46                         <im>
1340 nextz:nexts                        <fb>
1350 return                             <ce>
1360 :                                  <oe>
1370 rem horizontalspiegeln             <hi>
1380 fors=0to3                          <gi>
1390 forz=0to7                          <ep>
1400 p1=3072+s*40+z:p2=3072+(7-s)*
40+z                                    <ce>
1410 pp1=peek(p1):pokep1,peek(p2):
pokep2,pp1                              <ce>
1420 nextz:nexts                        <go>
1430 return                             <mf>
1440 :                                  <if>
1450 rem vertikalspiegeln               <ap>
1460 fors=0to7                          <jc>
1470 forz=0to3                          <gp>
1480 p1=3072+s*40+z:p2=3072+s*40+7
-z                                      <hm>
1490 pp1=peek(p1):pokep1,peek(p2):
pokep2,pp1                              <mo>
1500 nextz:nexts                        <ge>
1510 return                             <gf>
1520 :                                  <cf>
1530 rem scroll                         <og>
1540 char1,25,16,"scroll:"              <bj>
1550 geta$                              <mn>
1560 ifa$=c2$thenchar1,33,16,b6$+l
eft$(ql$,6)+"hoch":sc=1                 <gd>
1570 ifa$=c4$thenchar1,33,16,b6$+l
eft$(ql$,6)+"runter":sc=2               <cm>
1580 ifa$=c3$thenchar1,33,16,b6$+l
eft$(ql$,6)+"rechts":sc=3               <md>
1590 ifa$=c1$thenchar1,33,16,b6$+l
eft$(ql$,6)+"links":sc=4                <gg>
1600 ifa$=chr$(13)thengoto1630          <ik>
1610 goto1550                           <il>
1620 :                                  <oo>
1630 onscgosub1660,1740,1820,1900       <gm>
1640 return                             <gk>
1650 :                                  <ck>
1660 rem hoch-scroll                    <fg>
1670 fors=1to7                          <gb>
1680 forz=0to7                          <ei>
1690 fort=0tog                          <hk>
```

```
1700 ifpeek(3072+s*40+z)=p(t)thenp
oke3072+s*40+z-40,p(t):poke3072+s*
40+z,46                           <am>
1710 nextt:nextz:nexts            <el>
1720 return                       <al>
1730 :                            <ml>
1740 rem runter-scroll            <li>
1750 fors=6to0step-1              <pl>
1760 forz=0to7                    <dm>
1770 fort=0to6                    <ff>
1780 ifpeek(3072+s*40+z)=p(t)thenp
oke3072+s*40+z+40,p(t):poke3072+s*
40+z,46                           <cm>
1790 nextt:nextz:nexts            <ll>
1800 return                       <km>
1810 :                            <gm>
1820 rem rechts-scroll            <ld>
1830 ford=0tomf                   <ip>
1840 fors=0to7:forz=6to0step-1    <ak>
1850 fort=1tog                    <ig>
1860 ifpeek(3072+s*40+z)=p(t)thenp
oke3072+s*40+z+1,p(t):poke3072+s*4
0+z,46                            <lp>
1870 nextt,z,s,d                  <jn>
1880 return                       <em>
1890 :                            <am>
1900 rem links-scroll             <pg>
1910 ford=0tomf                   <fd>
1920 fors=0to7:forz=1to7step1     <pg>
1930 fort=0tog                    <lm>
1940 ifpeek(3072+s*40+z)=p(t)thenp
oke3072+s*40+z-1,p(t):poke3072+s*4
0+z,46                            <ko>
1950 nextt,z,s,d                  <ei>
1960 return                       <on>
1970 :                            <kn>
1980 rem drehung                  <bf>
1990 char1,11,16,"drehung:"       <cg>
2000 geta$                        <oh>
2010 ifa$="1"thenchar1,20,16,b3$+c
1$+c1$+c1$+" 90":dr=1             <ph>
2020 ifa$="2"thenchar1,20,16,b3$+c
1$+c1$+c1$+"180":dr=2             <mh>
2030 ifa$="3"thenchar1,20,16,b3$+c
1$+c1$+c1$+"270":dr=3             <id>
2040 ifa$=chr$(13)thengoto2070    <fp>
2050 goto2000                     <li>
2060 :                            <gc>
2070 fors=0to7                    <lj>
2080 forz=0to7                    <ko>
2090 d(s*8+z)=peek(3072+s*40+z)   <np>
2100 poke3072+s*40+z,46           <pl>
2110 nextz:nexts                  <pa>
2120 :                            <nk>
2130 ondrgosub2150,2230,2310      <oa>
2140 return                       <fg>
2150 rem  90 grad                 <ck>
2160 fors=0to7                    <bp>
2170 forz=0to7                    <ed>
2180 p2=3072+(z)*40+7-s           <fm>
2190 fort=0tog                    <ph>
2200 ifd(s*8+z)=p(t)thenpokep2,p(t
)                                 <cf>
2210 nextt:nextz:nexts            <db>
2220 return                       <ph>
2230 rem 180 grad                 <ke>
2240 fors=0to3                    <ea>
2250 forz=0to7                    <dl>
2260 fort=0tog                    <gj>
2270 ifd(s*8+z)=p(t)thenpoke3072+(
7-s)*40+7-z,p(t)                  <pl>
2280 ifd((7-s)*8+7-z)=p(t)thenpoke
3072+s*40+z,p(t)                  <il>
2290 nextt:nextz:nexts            <aj>
2300 return                       <jg>
2310 rem 270 grad                 <jd>
2320 fors=0to7                    <hi>
2330 forz=0to7                    <gl>
2340 p2=3072+(7-z)*40+s           <hp>
2350 fort=0tog                    <ap>
2360 ifd(s*8+z)=p(t)thenpokep2,p(t
)                                 <oa>
2370 nextt:nextz:nexts            <ja>
2380 return                       <di>
2390 :                            <pi>
2400 rem codewahl                 <lp>
2410 :                            <ca>
2420 geta$                        <be>
2430 ifa$="+"thenzc=zc+1:ifzc>127t
henzc=0                           <lh>
2440 ifa$="-"thenzc=zc-1:ifzc<0the
nzc=127                           <mf>
2450 ifa$="f"thengoto2800         <fg>
2460 ifa$=chr$(13)then2520        <ig>
2470 printhe$left$(qd$,12)" "rn$"c
ode"rf$":"using"###";zc;          <pj>
2480 char1,1,14,b6$+" "+rn$+"[+/-]
"+rf$+" ":poke3562,zc             <jb>
2490 goto2420                     <ph>
2500 return                       <cj>
2510 :                            <oj>
2520 rem zeichen vergroessern     <gd>
2530 :                            <bb>
2540 ad=dec("3800")               <jg>
2550 fors=0to7                    <bb>
2560 forz=mfto7+mfstep1+mf        <cj>
2570 p1=(peek(ad+zc*8+s)and2^(7-z)
):ifmf=1thenp2=(peek(ad+zc*8+s)and
2^(7-z+1)):elsep2=p1              <fe>
2580 p=3072+z+s*40                <bn>
2590 ifp1<>2^(7-z)thenifmf=0thengo
to2640:elseifmf=1then2610         <ll>
2600 ifp1=2^(7-z)thenifmf=0thenpok
ep,81:goto2650:else2610           <nd>
2610 ifp1=1*2^(7-z)andp2=1*2^(7-z+
1)thenpokep,81:pokep-1,81:goto2650 <ee>
```

```
2620 ifp1=1*2^(7-z)andp2=0thenpoke
p,87:pokep-1,87:goto2650          <kp>
2630 ifp2=1*2^(7-z+1)andp1=0thenpo
kep,42:pokep-1,42:goto2650        <gp>
2640 pokep,46:ifmf=1thenpokep-1,46 <am>
2650 next:next                    <bp>
2660 goto2500                     <ej>
2670 :                            <ck>
2680 rem altes zeichen zurueck    <en>
2690 :                            <fd>
2700 ad=dec("d000")               <ig>
2710 poke1177,62                  <eh>
2720 fors=0to7                    <ng>
2730 forz=0to7                    <lp>
2740 p=(peek(ad+zc*8+s)and2^(7-z)) <df>
2750 ifp=0thenchar1,z,s,".":elsech
ar1,z,s,"Q"                       <gl>
2760 next:next                    <jd>
2770 poke1177,63                  <bd>
2780 return                       <fl>
2790 :                            <bl>
2800 rem zeichen uebernehmen      <me>
2810 :                            <ec>
2820 fors=0to7:p=0                <ho>
2830 forz=7tomfstep-1-mf          <ho>
2840 p1=3072+z+s*40:ifmf=1thenp2=3
072+z+s*40-1:elsep2=p1            <jo>
2850 ifpeek(p1)=81orpeek(p2)=81the
np=p+2^(7-z):ifmf=1thenp=p+2^(7-z+
1)                                <pc>
2860 ifpeek(p1)=87orpeek(p2)=87the
np=p+2^(7-z)                      <hj>
2870 ifpeek(p1)=42orpeek(p2)=42the
np=p+2^(7-z+1)                    <cj>
2880 next:pokedec("3800")+zc*8+s,p
:next                             <fe>
2890 z%(zc)=1:flag=1:return       <oc>
2900 :                            <pi>
2910 rem zeichen in datas ablegen <ih>
2920 :                            <ca>
2930 ifflag=0thengoto360          <jk>
2940 k=0:fort=0to127:poke1383+t,z%
(t):next                          <kh>
2950 ifpeek(1383+k)=1thenad=dec("3
800"):zd=zd+10:goto3000:elsek=k+1 <ld>
2960 ifk<127thengoto2950          <bk>
2970 goto3080                     <eg>
2980 gosub3340:goto360            <aj>
2990 :                            <kn>
3000 gosub3340:printcl$c4$c4$c4$zd
"data"k",";                       <hl>
3010 fort=0to7                    <jn>
3020 d=peek(ad+k*8+t):d$=str$(d)  <ef>
3030 printright$(d$,len(d$)-1)",";
:next:printc1$" "                 <ge>
3040 gosub3120:print"goto2950"chr$
(19)                              <ce>
3050 :                            <cf>
3060 poke1319,13:poke1320,13:poke2
39,2:end                          <kj>
3070 :                            <em>
3080 gosub3340:printcl$c4$c4$c4$zd
+10"data -1"                      <ih>
3090 gosub3120:print"goto2980"chr$
(19)                              <ab>
3100 :                            <ik>
3110 poke1319,13:poke1320,13:poke2
39,2:end                          <jp>
3120 print"zd="zd":po="po":k="k+1"
:dimz%(127):";:return             <gk>
3130 rem normal/multicolor-zeichen <il>
3140 :                            <nk>
3150 scnclr                       <jm>
3160 print" aktueller modus : "mf$
(mf)                              <fj>
3170 print:print                  <ob>
3180 print" soll das zeichen in "r
n$"m"rf$"ulticolor oder "         <ip>
3190 print" "rn$"n"rf$"ormal erste
llt werden ?"                     <oj>
3200 print:print:print            <ha>
3210 print" QQ = zeichenfarbe 1   <bf>
3220 print" WW = zeichenfarbe 2   <gb>
3230 print" ** = zeichenfarbe 3   <de>
3240 getkeym$                     <pc>
3250 ifm$="m"thenmf=1:g=3:x=2*int(
x/2):b=2:goto370                  <oc>
3260 ifmf=1then370:else:mf=0:b=1:g
=1:goto370                        <hi>
3270 :                            <np>
3280 rem ende                     <ld>
3290 :                            <ah>
3300 end                          <gb>
3310 :                            <cp>
3320 rem nachspann ============== <do>
3330 rem * farbcodes/steuercodes * <hb>
3340 c4$=chr$(017):rn$=chr$(018)  <ac>
3350 he$=chr$(019):re$=chr$(028)  <jk>
3360 c3$=chr$(029):bk$=chr$(144)  <ai>
3370 c2$=chr$(145):rf$=chr$(146)  <dc>
3380 cl$=chr$(147):c1$=chr$(157)  <ph>
3390 rem *** zeichensatz/graphik * <kh>
3400 zs$=chr$(189):ym$=chr$(219)  <lo>
3410 yo$=chr$(221)                <pe>
3420 rem ********* zeichenfolgen * <ka>
3430 qd$="":qu$="":ql$="":forq=1to
40                                <ep>
3440 qd$=qd$+c4$:qu$=qu$+c2$      <lp>
3450 ql$=ql$+c1$                  <bf>
3460 nextq:b$=chr$(32):b2$=b$+b$  <cd>
3470 b3$=b2$+b$:b6$=b3$+b3$       <en>
3480 b9$=b6$+b3$:b$=b9$+b$        <dn>
3490 return                       <on>
3500 rem ======================== <de>
8000 rem einleseprogramm          <lk>
8010 :                            <am>
```

```
8020 readz:ifz=-1thenpoke65298,0:p
oke65299,56:goto65535              <op>
8030 fort=0to7:readd:poke14336+z*8
+t,d:next:goto65535                <nb>
8040 :                             <ej>
8050 rem datas fuer neue zeichen   <pg>
8060 :                             <hb>
8070 =p=r=o=g=r=a=m=m=e=n=d=e===== <fg>
```

Kalender 2300!

Dies ist kein Kalenderprogramm für das Jahr 1988 oder 1989. Das Programm Kalenderrechnung kennt die Daten vom Jahr 1 bis zum Jahr 2299. Wer wissen will, wie der Kalender errechnet wird, findet hier die nötigen Informationen.

Warum fängt eigentlich nicht jedes Jahr mit einem Sonntag an; warum fallen die Festtage nicht jedes Jahr auf den gleichen Wochentag?
Es liegt am Kalender. An unserem Kalender der christlichen Zeitrechnung, wobei das „christlich" wichtig ist. Denn alle gebräuchlichen Kalender sind entstanden aus Religion und Naturwissenschaft, aus den religiösen Vorstellungen der Menschen.
Noch heute deutlich wird dieser uralte Zusammenhang in der Astrologie, denn die Einteilung des Jahres in die zwölf magischen Sternkreiszeichen ist letztlich auch nichts anderes als ein Kalender, dessen Voraussetzung, nämlich das Eintreten bestimmter Sterne in einen Sektor des sichtbaren Himmels zu einem festen Zeitpunkt, schon längst nicht mehr zutrifft. Denn die Bewegungen der Erde, des Sonnensystems und unserer Galaxis haben im Laufe der Jahrtausende die Konstellationen weit verschoben.
Der Kalender stimmt nicht mehr. Und das ist etwas, das der astrologische Kalender über kurz oder lang mit allen anderen gemein hat. Sie müssen immer wieder korrigiert werden. Wie das bei unserem eigenen Kalender funktioniert, wie er entstanden ist, wie man ihn errechnet, das soll hier näher betrachtet werden.
Dienstag, 25. August 1987. Ein ganz normales Datum. Es besteht aus drei Teilen: dem Tag, nochmals unterteilt in Wochentag und Tagesdatum, dem Monat und dem Jahr. Die einfachste Einheit ist der Tag, nämlich eine Umdrehung der Erde um sich selbst.
Schauen wir als nächstes das Jahr an. Auch da können wir mit unseren Sinnen den Ablauf erfahren. Immer wenn es wieder Frühling wird, sind wir ein Jahr weiter. Wissenschaftlich also die Zeitspanne, in der die Erde einmal die Sonne umkreist.
Dann der Monat. Er hat den direkten Zusammenhang mit den Bewegungen eines Himmelskörpers ein wenig verloren. Theoretisch ist ein sogenannter synodischer Mondmonat die Zeit, in der der Mond den vollen Phasenzyklus vom Neumond zum Vollmond und zurück durchläuft. Dieser Mondkalender ist aber schon lange bis auf bestimmte Reste nicht mehr im Gebrauch. Ein Monat ist für uns einfach die Unterteilung eines Jahres in überschaubare, etwa gleichlange Einheiten.
Dienstag, 25. August 1987. Ein Datum, beruhend auf den Bewegungen dreier Himmelskörper; der Erde, des Mondes und der Sonne. Und die Religion? Sie spielt eine Rolle, und zwar gleich in zweifacher Hinsicht: Zum einen gab sie den Anstoß zur Errechnung von Kalendern. Eine Menschheit, die Naturgottheiten verehrte, konnte, solange sie als Bauern in Dörfern selbst abhängig von der Natur lebte, ohne festen Kalender auskommen. Die Göttin der Fruchtbarkeit wurde zur Zeit des Frühjahrsvollmondes verehrt, und Frühjahr war es, wenn der Schnee schmolz.
Sobald aber Städte entstanden waren, mit ihrer fortgeschrittenen Arbeitsteilung, mit der Aufteilung der Menschen in Klassen – Sklaven, Bauern, Handwerker und Händler, Krieger, Könige und Priesterkönige –, da hatte man sich ein Stück von der Natur entfernt. Man lebte auch nicht mehr nach einem einheitlichen Rhythmus und es wurde notwendig, dem Leben eine feste zeitliche Einteilung zu geben, Feiertage festzulegen.
In Mesopotamien und Ägypten entwickelte sich die Astronomie und parallel dazu ihre mystisch-magische Zwillingsschwester, die Astrologie. Es war nicht nur die praktische Notwendigkeit, Himmelserscheinungen vorauszuberechnen, um Feiertage festzulegen oder den Zeitpunkt des Nilhochwassers zu bestimmen, sondern es war natürlich auch beeindruckend für die Massen, wenn eine Zeremonie abgehalten wurde, der Priester den Arm hob und die Sonne sich verfinsterte.
Sehr früh auch entstand ein Teil unseres Kalenders, der noch heute unser aller Leben prägt, die Woche. Feiertage kannten die Babylonier, die Ägypter, die Griechen und Römer. Es waren die Juden, die diese Feiertage in einen festen, sich stetig wiederholenden Rhythmus brachten. „Der siebte Tag ist ein Ruhetag, dem Herrn, deinem Gott, geweiht." (Ex 20,10)
Die Zehn Gebote haben uns also unseren siebentägigen Wochenrhythmus gebracht.
Wie sind diese verschiedenen Bestandteile in Einklang zu bringen? Schlecht. Denn die Erde umkreist die Sonne in 365,24219879 Tagen. Der Phasenzyklus des Mondes dauert 29,530589 Tage. Damit kann man im täglichen Leben nicht rechnen. Also gibt es in Kalendern Normal- oder Gemeinjahre, und es gibt Jahre, die verlängert werden durch das Einschalten von Extratagen. Durch diese Konstruktionen versuchte man, sich der tatsächlichen Jahreslänge so weit wie möglich anzunähern.
Unser Kalender beruht auf dem der Römer. Das römische Jahr hatte ursprünglich zehn Monate, von März bis Dezember (decem = 10). Recht früh kamen Januar und Februar dazu. Die Länge des Jahres betrug allerding nur 355 Tage. Diese Ungenauigkeit suchte man durch ganze Serien von Schalttagen oder Schaltmonaten auszugleichen. Eine ungeheuer komplizierte und auch politisch sehr wichtige Berechnung, denn in Rom amtierten die Beamten einschließlich der Konsuln, die die höchste zivile und militärische Gewalt hatten, nur ein Jahr.
Die neuen Jahresbeamten sollten ihr Amt jeweils am Jahresanfang antreten, um die zivilen Angelegenheiten ordnen zu können, das Heer vorzubereiten und dann für die Sommerfeldzüge bereit zu sein. Das konnte natürlich nicht gutgehen, wenn das Jahr immer später anfing, wenn auch immer öfter die Kompliziertheit der Kalenderrechnung ausgenutzt wurde von bestechlichen Priestern, die zugunsten der scheidenden Beamten das Jahr verlängerten oder es, falls die neuen mehr boten, arg verkürzten.
So regiert man keine Weltmacht. Und es brauchte dann auch einen Diktator, Gaius Iulius Caesar, der

dem Kalender reformierte (46 v. Chr.), indem er die Länge des Jahres auf 365 Tage festlegte. Alle vier Jahre wurde ein Schaltjahr von 366 Tagen eingelegt. Das war bereits recht genau, und es ist im wesentlichen unser heutiger Kalender. Die Abweichung vom tatsächlichen Sonnenjahr betrug in fast 1600 Jahren nur zehn Tage.

DER JULIANISCHE KALENDER

Dieser auf dem Sonnenjahr basierende julianische Kalender wurde dann im Mittelalter mit der jüdisch-christlichen Siebentagewoche in Einklang gebracht und man mußte ihn mit dem Mondumlauf synchronisieren, denn das Osterfest wurde nach dem Mondkalender berechnet. Man könnte fragen, ob das so wichtig war. Schließlich hätte man doch sagen können, Ostern sei jedes Jahr am 10. April.
Aber das war unmöglich, denn die Religion hatte im Mittelalter eine Bedeutung, die wir uns heute nicht mehr vorstellen können. Der Glaube bestimmte das ganze Leben, jeden Tag. Christus und die Kirche waren das Zentrum des Denkens. Könige und Adlige schenkten für ihr Seelenheil der Kirche, speziell den Klöstern, große Besitzungen. Das war kein billiger Ablaßhandel, sondern tiefe Überzeugung.
Im Hochmittelalter machten sich Städte von nur 20.000 Einwohnern daran, riesige Kathedralen zu bauen. Zum Teil mit den Händen schleppten die Bürger die Steine, reiche Kaufleute trugen Balken, karrten Sand. Nur aus Rivalität mit der Nachbarstadt baut man nicht hundert Jahre lang, sondern man baut zur höheren Ehre Gottes. Und wenn das so ist, dann kann man natürlich auch nicht einfach das Datum der Auferstehung Christi der Bequemlichkeit halber verschieben. Wenn es heute vielleicht möglich wäre, Ostern als einen Gedenktag zu sehen, der ein festes Datum erhält, so war es für das Mittelalter selbstverständlich, daß man lieber komplizierte Berechnungen durchführt, um das Sonnenjahr mit dem Mondjahr zu synchronisieren, als daß man den heiligen Ostertermin falsch festlegt.

DIE GOLDENE ZAHL

Und so errechnete man den Mondzyklus, der in 19 julianischen Jahren 235 synodische Mondmonate enthielt. Das erforderte wieder Mondschaltjahre und Mondschalttage, war kompliziert und umständlich. Aber man hatte einen immer wiederkehrenden Rhythmus von 19 Jahren, der die ganze christliche Zeitrechnung durchläuft.
Um die Stellung eines Jahres im Mondzyklus zu bestimmen, gibt es die „Goldene Zahl". Sie wird recht einfach ermittelt, indem man zur Jahreszahl Eins addiert, denn der Mondzyklus knüpft an das Jahr 1 v. Chr. an, und das Ergebnis durch 19 teilt. Der Rest ist die Goldene Zahl. Geht die Rechnung glatt auf, dann ist die Goldene Zahl 19.
Um dies zur Festrechnung zu benutzen, fehlt uns noch der Sonnenzyklus. Er dient dazu, das julianische Jahr mit der siebentägigen Woche zu verbinden. Nehmen wir einmal an, das Jahr hätte 364 Tage. Das wären genau 52 Wochen. Jedes Jahr finge mit dem gleichen Tag an, vermutlich einem Sonntag, und wir bräuchten überhaupt keinen Kalender, weil alles so schön regelmäßig wäre. Wenn wir nun ein Jahr von 365 Tagen annehmen, dann finge zum Beispiel das Jahr 1 mit einem Sonntag an, das Jahr 2 mit einem Montag, und nach einem siebenjährigen Zyklus wären wir wieder beim Sonntag. Weil aber außerdem alle vier Jahre ein Schalttag eingelegt wird, so dauert es 4*7=28 Jahre, bis die Wochentage wieder auf dieselben Monatsdaten fallen. Der Sonnenzyklus ist also 28 Jahre lang.

ERRECHNEN DER WOCHENTAGE

Um Wochentage bestimmen zu können, teilte man jedem Tag des Jahres einen der Buchstaben von A bis G zu, teilte das Jahr also in Wochen und zwar ganz einfach und ein für allemal festgelegt. 1. Januar = A, 2. = B, 8. Januar wieder = A und so weiter bis zum Jahresende. Die Jahre des Sonnenzyklus wurden mit dem Buchstaben benannt, den der erste Sonntag im Jahr hatte, dem „Sonntagsbuchstaben".
Um einen Tag zu bestimmen, sah man in einer Tabelle nach, welchen Tagesbuchstaben dieses Datum hatte. Das wäre für unseren 25. August „F". Das Jahr 1987 hat den Sonntagsbuchstaben „D". „F" kommt im Alphabet zwei Stellen nach „D". Wenn also „D" ein Sonntag ist, dann ist jeder Tag in diesem Jahr ein Dienstag.
Um den Sonntagsbuchstaben des Jahres zu ermitteln, mußte man natürlich wissen, an welcher Stelle im Sonnenzyklus das Jahr stand. Das Jahr 1 des Zyklus ist ein Schaltjahr, das mit einem Montag beginnt, so auch das Jahr 9 v. Chr. Die Rechnung ist ähnlich wie bei der Goldenen Zahl; Jahreszahl plus 9 durch 28. Der Rest, oder falls kein Rest bleibt, 28, ist die Zahl des Sonnenzyklus, für die man in einer Tabelle den Sonntagsbuchstaben suchen konnte. Schaltjahre hatten zwei davon. Der erste galt bis zum 28. Februar, dann folgte ein Schalttag. Also verschoben sich die Wochentage und es ergab sich ein neuer, fiktiver, erster Sonntag im Jahr. Daher galt vom 1. März an der nächste Sonntagsbuchstabe.

OSTERBERECHNUNG

Wie berechnet man den Termin des Osterfestes? Christus wurde am Rüsttag des Passahfestes gekreuzigt, das am Tag des ersten Vollmondes nach Frühlingsanfang stattfand. Er stand auf von den Toten am dritten Tag, dem ersten Tag der neuen Woche, der deshalb zum Tag des Herrn wurde, dem Sonntag, und im Christentum den Sabbath als wöchentlichen Festtag ablöste.
Im Jahre 325 legte das Konzil von Nikäa fest, daß Ostern am ersten Sonntag nach dem Frühjahrsvollmond, der sogenannten Ostergrenze, gefeiert werden sollte. Der Frühjahrsbeginn (Frühjahrs-Tag- und Nachtgleiche) wurde wegen der Schwierigkeit der astronomischen Berechnung auf den 21. März festgelegt.
Zur Osterberechnung braucht man also den 28jährigen Sonnenzyklus, um den Wochentag, und den 19jährigen Mondzyklus, um den Vollmond zu berechnen. Im 6. Jahrhundert schuf der römische Abt Dionysius Exiguus aus der Kombination beider den 532jährigen Osterzyklus und gab in Tabellen für jedes Jahr des Mondzyklus die Ostergrenze an.
Man errechnet also mit Hilfe der Goldenen Zahl die Stellung des Jahres im Mondzyklus, schlägt in einer Tabelle die Ostergrenze und ihren Tagesbuchstaben nach und berechnet dann über den Sonnenzyklus und den Sonntagsbuchstaben das Datum des auf die Ostergrenze folgenden Sonntags.

Lesen Sie bitte weiter auf Seite 55

```
10 rem kalender=================p4   <oa>
20 rem (p) commodore welt team       <ho>
30 rem ==========================    <ng>
40 rem (c) by fwk                    <kb>
50 rem          wiesloch             <if>
60 rem basic v3.5                    <od>
70 rem plus4 (c16/116 + 64k)         <ge>
80 rem drucker optional              <ec>
90 rem (citizen 120-d)               <kn>
100 rem ==========================   <id>
110 graphic1,1:clr:graphic 0,1:gos
ub4320                               <dh>
120 scnclr: char1,11,11,"bitte war
ten !"                               <ki>
130 rem ==========================   <np>
140 rem variable dimensionieren      <cn>
150 rem ==========================   <nj>
160 dim mo$(12),ta$(7),mc(13),sb(2
8),gj(7),zs(7),og(7,19)              <hl>
170 dim f1$(32),f1(32),f2(32),f3(3
2),v2(32)                            <bn>
180 dim h1$(16),h1(16),h3(16)        <ik>
190 rem ==========================   <oh>
200 rem datazeilen                   <gg>
210 rem ==========================   <en>
220 rem monatsnamen   = mo$(12)      <ec>
230 rem ==========================   <jc>
240 data januar,februar,maerz,apri
l,mai,juni                           <ok>
250 data juli,august,september,okt
ober,november,dezember               <af>
260 rem ==========================   <gm>
270 rem tagesnamen   = ta$(7)        <gj>
280 rem ==========================   <lo>
290 datasonntag,montag,dienstag,mi
ttwoch                               <kg>
300 data donnerstag,freitag,samsta
g                                    <ph>
310 rem ==========================   <hp>
320 rem monatskonstante   = mc(13)   <hc>
330 rem ==========================   <mn>
340 data 0,31,59,90,120,151,181      <df>
350 data 212,243,273,304,334,365     <nd>
360 rem ==========================   <bn>
370 rem sonntags 'buchstaben'   = s
b(28)                                <eg>
380 rem ==========================   <da>
390 data 76,5,4,3,21,7,6,5,43,2,1,
7,65,4                               <ci>
400 data 3,2,17,6,5,4,32,1,7,6,54,
3,2,1                                <of>
410 rem ==========================   <bb>
420 rem feste sonntagsbuchstaben =
 fb(5)                               <de>
430 rem ==========================   <im>
440 data 3,5,7,3,5                   <ph>
450 rem ==========================   <do>
460 rem grenzjahre   = gj(7)         <ln>
470 rem ==========================   <nd>
480 data 15821004,17000101,1800010
1,19000101,21000101,22000101,1e20    <eb>
490 rem ==========================   <pm>
500 rem zyklusstart   = zs(7)        <if>
510 rem ==========================   <af>
520 data-9,1567,1691,1787,1883,209
1,2187                               <ml>
530 rem ==========================   <bm>
540 rem ostergrenzen   = og(7,19)    <df>
550 rem ==========================   <pp>
560 rem og(1)                        <mj>
570 rem ==========================   <mo>
580 data 15,4,23,12,1,20,9,28,17,6   <ck>
590 data 25,14,3,22,11,0,19,8,27     <bd>
600 rem ==========================   <km>
610 rem og(2)                        <ea>
620 rem ==========================   <ob>
630 data 22,11,0,19,8,27,16,5,24,1
3                                    <ii>
640 data 2,21,10,28,18,7,26,15,4     <ij>
650 rem ==========================   <ge>
660 rem og(3+4)                      <hd>
670 rem ==========================   <fh>
680 data 23,12,1,20,9,28,17,6,25,1
4                                    <ji>
690 data 3,22,11,0,19,8,27,16,5      <im>
700 rem ==========================   <oh>
710 rem og(5+6)                      <db>
720 rem ==========================   <en>
730 data 24,13,2,21,10,28,18,7,26,
15                                   <nf>
740 data 4,23,12,1,20,9,27,17,6      <bl>
750 rem ==========================   <bf>
760 rem og(7)                        <pd>
770 rem ==========================   <gm>
780 data 25,14,3,22,11,0,19,8,27,1
6                                    <ep>
790 data 5,24,13,2,21,10,28,18,7     <ga>
800 rem ==========================   <bk>
810 rem daten der festtage           <bb>
820 rem ==========================   <hp>
830 data ostern,0,1,3,ostern,1,2,3
,karfreitag,-3,6,3                   <ig>
840 data christi himmelf.,41,5,5,p
fingsten,10,1,5                      <md>
850 data pfingsten,1,2,5,fronleich
nam,10,5,5                           <kb>
860 data 1,1,neujahr,6,1,hl.drei k
oenige,1,5,tag der arbeit            <kl>
870 data 17,6,t.d.dtsch.einh.,15,8
,mariae himmelf.                     <no>
880 data 1,11,allerheiligen,25,12,
weihnachten,26,12,weihnachten,buss
und bettag                           <ja>
890 rem ==========================   <da>
900 rem data lesen                   <gc>
910 rem ==========================   <ic>
```

```
920 for x=1 to 12: read mo$(x): ne
xt x                                 <ae>
930 for x=1 to 7: read ta$(x):next
 x                                   <il>
940 for x=1 to 13: read mc(x): nex
t x                                  <pi>
950 for x=1 to 28: read sb(x):next
 x                                   <lj>
960 for x=2 to 6: read fb(x): next
 x                                   <jn>
970 for x=1 to 7: read gj(x): next
 x                                   <ma>
980 for x=1 to 7: read zs(x): next
 x                                   <ah>
990 for x=1 to 7                     <nm>
1000 if x=4 then restore 680         <hf>
1010 if x=6 then restore 730         <jp>
1020 for y=1 to 19: read og(x,y):
next y                               <pc>
1030 next x                          <if>
1040 rem ========================= <op>
1050 rem textstrings                 <fm>
1060 rem ========================= <gb>
1070 t1$=b4$+b4$+"*** programm kal
ender ***"+b4$+b3$                   <nd>
1080 t2$=b6$+"** weiter mit <space
> **"+b6$                            <mm>
1090 t3$=chr$(18)+t2$+chr$(146)      <fi>
1100 t4$=b2$+"festtagskalender des
 jahres"                             <pb>
1110 t5$="nr. d. wochentages ="      <od>
1120 t6$="tagesdatum"+b5$+b4$+"="    <kb>
1130 t7$="nr. d. monats"+b6$+"="     <id>
1140 t8$="jahr"+b$+b5$+"="           <fb>
1150 t9$="zusatzeingaben"            <ec>
1160 t0$=chr$(145)                   <nb>
1170 l1$="": for x=1 to 40: l1$=l1
$+"-": next x                        <fo>
1180 l2$="": for x=1 to 46: l2$=l2
$+"*": next x                        <be>
1190 rem ========================= <pc>
1200 rem windowstrings               <na>
1210 rem ========================= <ff>
1220 w1$=chr$(18)+t1$+chr$(146)+ch
r$(27)+"t"                           <fh>
1230 w2$=chr$(27)+"n"                <lo>
1240 w3$=chr$(27)+"t"                <ap>
1250 w4$=chr$(27)+"q":w5$=chr$(18)   <ho>
1260 w5$=w5$+"bezeichnung(max.16)
:"+b2$+"tag"+b4$+"monat [nr]"+chr$
(146)+chr$(27)+"t"                   <gk>
1270 rem ========================= <cm>
1280 rem druckersteuerung            <ej>
1290 rem ========================= <dl>
1300 rem master reset                <ei>
1310 p1$=chr$(27)+"@"                <do>
1320 rem linken rand setzen in spa
lte 10                               <cn>
1330 p2$=chr$(27)+chr$(108)+chr$(1
0)                                   <hp>
1340 rem var. htabs in spalten 20,
25,36                                <fp>
1350 p3$=chr$(27)+"d"+chr$(20)+chr
$(25)+chr$(36)+chr$(0)               <dc>
1360 rem vertikale vergroesserung
ein                                  <oh>
1370 p4$=chr$(27)+chr$(126)+chr$(4
9)+chr$(49)                          <ak>
1380 rem vertikale vergroesserung
aus                                  <ic>
1390 p5$=chr$(27)+chr$(126)+chr$(4
9)+chr$(48)                          <in>
1400 rem sprung zum naechsten htab <jk>
1410 p6$=chr$(9)                     <dn>
1420 rem ========================= <kc>
1430 rem eroeffnungsgrafik           <pd>
1440 rem ========================= <kp>
1450 graphic 1,1                     <di>
1460 for x=1 to 100 step 5           <ne>
1470 box1,160-x,100-x,160+x,50+x     <mc>
1480 next x                          <jc>
1490 box0,125,85,195,115,,1          <on>
1500 char1,16,12,"kalender"          <ag>
1510 char1,2,24,t2$,1                <na>
1520 getkey x$: graphic 0            <lf>
1530 rem ========================= <ge>
1540 rem menue                       <ji>
1550 rem ========================= <op>
1560 print w2$;w1$: char1,5,3,w3$    <ne>
1570 x=0: y=0: f=0: x1=0: x2=0: x5
=0                                   <jb>
1580 print "bitte geben sie die nu
mmer des"                            <jj>
1590 print "gewuenschten unterprog
ramms ein:": print: print            <jn>
1600 print "wochentag = 1": print    <md>
1610 print "tagesdatum= 2": print    <ca>
1620 print "monat"b5$"= 3": print    <dl>
1630 print "festtage"b2$"= 4": pri
nt                                   <go>
1640 print "end"b6$" = 5"            <ij>
1650 trap 3760                       <ip>
1660 getkey x: on x gosub 1720,191
0,2220,2550,1680                     <nn>
1670 goto 1560                       <ca>
1680 print w2$: end                  <ga>
1690 rem ========================= <an>
1700 rem unterprogramm wochentagsu
che                                  <fk>
1710 rem ========================= <hi>
1720 scnclr: char1,8,3,w3$           <bp>
1730 print "gesucht: der wochentag
"                                    <ad>
1740 char1,8,5,w3$                   <ia>
1750 print t6$: print: print t7$:
print: print t8$                     <om>
```

```
1760 char1,30,5,w3$                        <am>
1770 input t:print:input m:print:i
nput a                                     <bk>
1780 gosub 3820: if f>0 then goto
3950                                       <pg>
1790 gosub 4080                            <fk>
1800 tt=mc(m)+t                            <pp>
1810 gosub 4260                            <ac>
1820 if x5=1 then return                   <pa>
1830 char1,8,12,w3$                        <ig>
1840 print "ergebnis:": print              <kd>
1850 print ta$(wz);",";tab(19) t;t
ab(23) mo$(m);tab(32) a                    <kn>
1860 char1,2,24,t3$: getkey x$             <gd>
1870 return                                <di>
1880 rem ========================          <pk>
1890 rem unterprogramm tagesdatum          <mb>
1900 rem ========================          <kd>
1910 scnclr: char1,0,3,w3$                 <je>
1920 for x=1 to 7: print left$(ta$
(x),2);"=";x: print: next x                <ok>
1930 char1,8,3,w3$                         <dn>
1940 print "gesucht: das tagesdatu
m"                                         <mb>
1950 char1,8,5,w3$                         <mn>
1960 print t5$: print: print t7$:
print: print t8$                           <nb>
1970 char1,30,5,w3$                        <fp>
1980 input wz: print: input m: pri
nt: input a: t=1                           <el>
1990 gosub 3820: if f>0 then goto
3950                                       <kl>
2000 gosub 4080                            <mg>
2010 tz=wz+sz-1: if tz>7 then tz=t
z-7                                        <hh>
2020 th=mc(m)+1: th=th-int(th/7)*7
: if th=0 then th=7                        <nd>
2030 if tz<th then tz=tz+7                 <fo>
2040 th=tz-th+1:x3=1:td(x3)=th:xh=
mc(3)                                      <kc>
2050 if x1=2 then mc(3)=mc(3)+1            <ml>
2060 do until mc(m)+td(x3)+6>=mc(m
+1)                                        <kj>
2070 x3=x3+1: td(x3)=td(x3-1)+7            <mg>
2080 loop                                  <dm>
2090 mc(3)=xh                              <ni>
2100 char1,8,12,w3$                        <ca>
2110 print "ergebnis:": print              <fh>
2120 z=1                                   <ia>
2130 do                                    <jp>
2140 print ta$(wz);",";tab(19) td(
z);tab(23) mo$(m);tab(32) a                <da>
2150 z=z+1                                 <oa>
2160 loop until z>x3                       <ne>
2170 char1,2,24,t3$: getkey x$             <ni>
2180 return                                <kh>
2190 rem ========================          <eo>
2200 rem unterprogramm monat               <ea>
2210 rem ========================          <pc>
2220 scnclr: char1,0,3,w3$                 <do>
2230 for x=1 to 7: print left$(ta$
(x),2);"=";x: print: next x                <kf>
2240 char1,8,3,w3$                         <db>
2250 print "gesucht: der monat"            <pn>
2260 char1,8,5,w3$                         <fi>
2270 print t5$: print: print t6$:
print: print t8$                           <ef>
2280 char1,30,5,w3$                        <dn>
2290 input wz: print: input t: pri
nt: input a: m=1                           <na>
2300 gosub 3820: if f>0 then goto
3950                                       <eg>
2310 gosub 4080                            <ln>
2320 x3=0                                  <le>
2330 for m=1 to 12                         <fp>
2340 if len(str$(x2))>2 and m=2 th
en xh=mc(3): mc(3)=mc(3)+1                 <ic>
2350 if len(str$(x2))>2 and m=3 th
en sz=val(mid$(str$(x2),3,1)): mc(
3)=xh                                      <gf>
2360 tz=wz+sz-1: if tz>7 then tz=t
z-7                                        <pk>
2370 th=mc(m)+t: if th>mc(m+1) the
n goto 2400                                <lc>
2380 th=th-int(th/7)*7:if th=0 the
n th=7                                     <hk>
2390 if th=tz then x3=x3+1: mh(x3)
=m                                         <jc>
2400 next m                                <pf>
2410 char1,8,12,w3$                        <fp>
2420 print "ergebnis:": print              <bc>
2430 if x3<>0 then goto 2460               <ap>
2440 print "im jahr";a;"gibt es ke
inen"                                      <hg>
2450 print ta$(wz);", den";t;".":
goto 2490                                  <me>
2460 for z=1 to x3                         <jd>
2470 print ta$(wz);",";tab(19) t;t
ab(23) mo$(mh(z));tab(32) a                <il>
2480 next z                                <ef>
2490 char1,2,24,t3$: getkey x$             <ad>
2500 return                                <cj>
2510 rem ========================          <el>
2520 rem unterprogramm festkalende
r                                          <fn>
2530 rem gesetzliche feiertage der
 brd                                       <oe>
2540 rem ========================          <ej>
2550 scnclr: char1,8,3,w3$                 <ck>
2560 print "gesucht: festkalender"
:print                                     <ic>
2570 print "rechenzeit ca. 10-45 s
ec"                                        <cb>
2580 char1,8,7,w3$                         <il>
2590 print t8$;:input a: print             <ij>
2600 input "zusatzeingaben (j/n)";
```

```
z1$: print                              <co>
2610 print "drucker oder"               <ll>
2620 print "bildschirm"b5$"(d/b)";
:inputz2$                               <dj>
2630 t=1: m=1                           <bj>
2640 gosub 3820: if f>0 then goto
3950                                    <ch>
2650 x5=1: x=0                          <gf>
2660 if z1$="n" then goto 2840          <ea>
2670 rem :                              <di>
2680 rem zusatzeingaben                 <oi>
2690 rem :                              <il>
2700 print w2$;t1$:print t9$;a:pri
nt w5$                                  <gg>
2710 do: x=x+1                          <el>
2720 printb2$;:inputh1$(x)              <dh>
2730 print t0$,,: input h1(x)           <ce>
2740 print t0$,,,: input h3(x): pr
int                                     <do>
2750 h1$(x)=left$(h1$(x),16)            <fl>
2760 t=h1(x): m=h3(x): gosub 3820       <ki>
2770 if f>0 then gosub 3950             <oj>
2780 ifx=16thenprint"keine weitere
n eingaben moeglich":char1,2,24,t3
$:getkey x$:exit                        <dm>
2790 input "weitere eingaben (j/n)
";z3$                                   <je>
2800 if z3$<>"j" then exit              <ag>
2810 print t0$;w4$;t0$;l1$              <nf>
2820 loop                               <ak>
2830 z6=x+16                            <ag>
2840 poke 65286,11                      <ln>
2850 for x=17 to z6                     <fo>
2860 f1$(x)=h1$(x-16): f1(x)=h1(x-
16): f3(x)=h3(x-16)                     <ci>
2870 t=f1(x): m=f3(x): gosub 3820       <hb>
2880 if f=5 then f1(x)=1: f3(x)=3:
 f=0: t=f1(x): m=f3(x)                  <jp>
2890 gosub 1790                         <hk>
2900 f2(x)=wz                           <db>
2910 next x                             <aa>
2920 rem :                              <ce>
2930 rem bewegliche feste               <go>
2940 rem :                              <hh>
2950 restore 830: m=3: gosub 4080       <li>
2960 gz=(a+1)-int((a+1)/19)*19          <af>
2970 tt=mc(3)+21+og(x4,gz)              <lc>
2980 gosub 4260                         <mn>
2990 for x6=1 to 7                      <fm>
3000 read f1$(x6),y,f2(x6),y1           <bc>
3010 tt=tt+y: t=tt-mc(y1): m=y1         <bf>
3020 if x6=1 then t=8-wz+t              <fk>
3030 if t+mc(y1)>mc(y1+1) then m=y
1+1: t=t-(mc(y1+1)-mc(y1))              <pc>
3040 if x6=1 then tt=t+mc(m)            <ol>
3050 f3(x6)=m: f1(x6)=t                 <nc>
3060 next x6                            <om>
3070 rem :                              <hn>
3080 rem feste feiertage                <of>
3090 rem :                              <nb>
3100 for x6=8 to 15                     <ki>
3110 read t,m,f1$(x6)                   <hg>
3120 f1(x6)=t: f3(x6)=m                 <df>
3130 gosub 1790                         <ji>
3140 f2(x6)=wz                          <fa>
3150 next x6                            <lp>
3160 rem :                              <oi>
3170 rem buss und bettag                <hl>
3180 rem :                              <di>
3190 x6=16: tt=mc(12)+25: gosub 42
60                                      <cb>
3200 if wz=1 then wz=8                  <lk>
3210 tt=tt-wz-31: t=tt-mc(11)           <lo>
3220 read f1$(x6): f1(x6)=t: f2(x6
)=4: f3(x6)=11                          <mh>
3230 rem :                              <ad>
3240 rem sortieren                      <kd>
3250 rem :                              <fc>
3260 if z6<x6 then z6=x6                <jk>
3270 for x=1 to z6                      <lm>
3280 mm$=str$(f3(x)): tt$="0"+mid$
(str$(f1(x)),2)                         <kh>
3290 tt$=right$(tt$,2): v2$=mm$+tt
$                                       <ib>
3300 v2(x)=val(v2$)                     <cj>
3310 next x                             <eg>
3320 y=0                                <bi>
3330 for x=1 to (z6-1)                  <pf>
3340 if v2(x+1)>=v2(x) then goto 3
410                                     <kj>
3350 c=f1(x): f1(x)=f1(x+1): f1(x+
1)=c                                    <ao>
3360 c=f2(x): f2(x)=f2(x+1): f2(x+
1)=c                                    <bd>
3370 c=f3(x): f3(x)=f3(x+1): f3(x+
1)=c                                    <ck>
3380 c$=f1$(x): f1$(x)=f1$(x+1): f
1$(x+1)=c$                              <ia>
3390 c=v2(x): v2(x)=v2(x+1): v2(x+
1)=c                                    <np>
3400 y=1                                <kc>
3410 next x                             <ni>
3420 if y>0 then goto 3320              <eo>
3430 poke 65286,27                      <ce>
3440 if z2$="d" then goto 3610          <ki>
3450 rem :                              <hh>
3460 rem ausgabe bildschirm             <bb>
3470 rem :                              <mh>
3480 print w2$;w1$: print t4$;a: p
rint                                    <ah>
3490 for x=1 to z6                      <fb>
3500 f3$=mo$(f3(x)): f2$=ta$(f2(x)
)                                       <od>
3510 if f1$(x)=f1$(x-1) then goto
3530                                    <kk>
3520 print f1$(x);                      <ee>
```

```
3530 printtab(16)f1(x);tab(20)f3$;
tab(30)f2$                              <ab>
3540 if x=16 and z6>16 then char1,
2,24,t3$: getkey x$: print w2$;w1$
: print t4$;a: print                    <on>
3550 next x                             <ak>
3560 char1,2,24,t3$: getkey x$          <hm>
3570 return                             <in>
3580 rem :                              <hn>
3590 rem ausgabe drucker                <lm>
3600 rem :                              <nb>
3610 open1,4,7: cmd1                    <gj>
3620 print p1$;p2$;p3$;l2$: print       <cp>
3630 print p4$;a;p5$: print             <ko>
3640 for x=1 to z6                      <gk>
3650 f3$=mo$(f3(x)): f2$=ta$(f2(x)
)                                       <pc>
3660 if f1$(x)=f1$(x-1) then goto
3680                                    <hl>
3670 print f1$(x);                      <jg>
3680 print p6$;f1(x);p6$;f3$;p6$;f
2$                                      <bf>
3690 next x                             <dm>
3700 print: print l2$                   <pf>
3710 print p1$: print#1: close1         <pm>
3720 return                             <ll>
3730 rem ========================       <an>
3740 rem subroutine trap-resume         <nc>
3750 rem ========================       <hi>
3760 char1,8,14,w3$                     <nm>
3770 print "sie haben einen fehler
 gemacht!"                              <fe>
3780 char1,2,24,t3$: getkey x$: re
sume 1560                               <ei>
3790 rem ========================       <jp>
3800 rem subroutine datumspruefung      <dp>
3810 rem ========================       <gm>
3820 if a=0 then f=1                    <lc>
3830 if a<0 or a>2299 then f=2          <pb>
3840 if t<=0 or t>31 then f=3           <bp>
3850 if m<=0 or m>12 then f=5: got
o 3910                                  <oh>
3860 if a>=1600 and right$(str$(a)
,2)="00" and a/400=int(a/400) and
m=2 then goto 3890                      <ji>
3870 if right$(str$(a),2)<>"00" an
d a/4=int(a/4) and m=2 then goto 3
890                                     <mj>
3880 if t>mc(m+1)-mc(m) then f=5        <mm>
3890 if t>(mc(m+1)-mc(m)+1) then f
=3                                      <nn>
3900 if a=1582 and m=10 and t>4 an
d t<15 then f=4                         <jh>
3910 return                             <di>
3920 rem ========================       <pk>
3930 rem subroutine fehlerangabe        <np>
3940 rem ========================       <kd>
3950 char1,8,14,w3$                     <ep>
3960 on f goto 3970,3980,3990,4000
,3990                                   <li>
3970 print "das jahr 'null' gibt e
s nicht!": goto 4020                    <cn>
3980 print "datum nicht im program
mbereich!": goto 4020                   <ph>
3990 print "dieses datum gibt es n
icht!": goto 4020                       <ng>
4000 print "kalenderumstellung -":
 print                                  <ec>
4010 print "dieses datum gibt es n
icht!"                                  <fa>
4020 char1,2,24,t3$: getkey x$: f=
0                                       <ko>
4030 if x5=1 then print w2$;w1$;t9
$;a: print w5$: x=x-1: f=0: return <mb>
4040 goto 1560                          <hm>
4050 rem ========================       <nf>
4060 rem subroutine sonntags'buchs
tabe'                                   <il>
4070 rem ========================       <ej>
4080 mm$="0"+mid$(str$(m),2)            <gb>
4090 tt$="0"+mid$(str$(t),2)            <ff>
4100 v1$=str$(a)+right$(mm$,2)+rig
ht$(tt$,2)                              <da>
4110 x4=0                               <ai>
4120 do: x4=x4+1                        <ch>
4130 if val(v1$)<gj(x4) then exit       <nd>
4140 if a<>1582 and a=val(left$(st
r$(gj(x4)),5)) then sz=fb(x4): x4=
x4+1: goto 4220                         <bc>
4150 loop                               <hi>
4160 r=a-zs(x4)                         <if>
4170 sz=r-int(r/28)*28                  <ci>
4180 if sz=0 then sz=28                 <ha>
4190 sz=sb(sz): x2=sz: if len(str$
(sz))<=2 then goto 4220                 <ib>
4200 if m<3 then x1=2: else x1=3        <ll>
4210 sz=val(mid$(str$(sz),x1,1))        <ah>
4220 return                             <kh>
4230 rem ========================       <eo>
4240 rem subroutine tages'buchstab
e'                                      <ee>
4250 rem ========================       <pc>
4260 tz=tt-int(tt/7)*7                  <ch>
4270 if tz=0 then tz=7                  <lo>
4280 if tz<sz then wz=tz+7: else w
z=tz                                    <nb>
4290 wz=wz-sz+1                         <di>
4300 return                             <eh>
4310 rem nachspann===============       <pe>
4320 b$=chr$(32):b2$=b$+b$              <ef>
4330 b3$=b2$+b$:b4$=b3$+b$              <od>
4340 b5$=b4$+b$:b6$=b5$+b$              <fk>
4350 b$=b5$+b5$:return                  <gb>
4360 rem ========================       <ma>
4370 rem 60671 bytes memory             <jp>
4380 rem 11788 bytes program            <fg>
```

```
4390 rem 00427 bytes variables        <dd>
4400 rem 02431 bytes arrays           <na>
4410 rem 01286 bytes strings          <bj>
4420 rem 12288 bytes graphic          <lj>
4430 rem 32451 bytes free (0)         <pd>
4440 rem ========================     <ha>
```

Kalender

Fortsetzung von Seite 49

DER GREGORIANISCHE KALENDER

So weit, so gut. Nur geschah es jetzt, daß durch die verbliebene Ungenauigkeit des julianischen Kalenders im Laufe der Jahrhunderte eine immer stärkere Abweichung vom Sonnenjahr eintrat. Im 16. Jahrhundert betrug sie bereits zehn Tage. Zwar ist das nicht viel, und ob das Frühjahr zehn Tage eher oder später anfängt, merkt bei unserem Wetter niemand.
Was aber sehr unangenehm war für eine religiöse Zeit, das war die Tatsache, daß offensichtlich der Termin des Osterfestes nicht mehr stimmte. Berechnet wurde es nach der Ostertafel des Dionysius. Tatsächlich aber geschah es, daß der Frühjahrsvollmond kam und ging, und Ostern wurde nicht gefeiert. Und hatte 46 v. Chr. Cäsar den Kalender wegen der Amtsjahre der römischen Beamten reformiert, so tat das jetzt im Jahre 1582 der Papst Gregor XIII., hauptsächlich wegen des Osterfestes.
Seine Reform korrigierte die im Lauf der Jahrhunderte entstandenen Ungenauigkeiten, indem man zehn Tage ausfallen ließ. Auf Donnerstag, den 4. Oktober 1582, folgte Freitag, der 15. Oktober 1582. Ferner wurde zur Verbesserung des Kalenders die Schaltregel geändert. Waren bisher alle durch vier teilbaren Jahre Schaltjahre, so wurde jetzt festgelegt, daß Säkularjahre, also die, die mit zwei Nullen enden, nicht durch vier, sondern durch vierhundert teilbar sein müssen, um Schaltjahre zu sein.
Das Jahr 1600 war also ein Schaltjahr, 1900 nicht. Dadurch fielen in 400 Jahren drei Tage aus, der Kalender wurde genauer. Weiter änderte man die Berechnung der Ostergrenzen, die jetzt komplizierter wurden, aber besser den Gestirnbewegungen angepaßt sind.

DAS PROGRAMM „KALENDER"

Das Programm „Kalender" arbeitet für die gesamte Dauer der christlichen Zeitrechnung. Das heißt, es berücksichtigt sowohl den julianischen als auch den gregorianischen Kalender. Es beginnt mit Datazeilen, von denen die ersten nicht erläutert werden müssen.
Die Monatskonstanten geben die bis zum Beginn des Monats vergangenen Tage an. Das ist wichtig für die Berechnung des Tagesdatums aus Sonntags- und Tagesbuchstaben, denn die Monate haben bekanntlich verschiedene Längen. Die Sonntagsbuchstaben entsprechen dem Sonnenzyklus, sind aber zur Berechnung durch Ziffern ersetzt.
Zur Vereinfachung wurden den Säkularjahren, die keine Schaltjahre sind, ihre Sonntagsbuchstaben fest als Data zugeteilt. Natürlich verschiebt sich durch diese Unregelmäßigkeit der Sonnenzyklus, so daß für die Zeit bis zur nächsten Unregelmäßigkeit ein neues Startjahr für die Berechnung der Sonnenzykluszahl festgelegt werden muß.
Es folgen die Ostergrenzen, die im 19jährigen Mondzyklus für sieben verschiedene Rechnungsperioden vom Jahr 1 bis 2299 den Termin des Ostervollmondes angeben – nicht als Datum, sondern als Abstand zum 21. März.
Zu den Zeilen 1200 bis 1260: Das Programm „Kalender" verzichtet bis auf „t0$" in Zeile 1160 auf Cursorsteuerung der Bildschirmausgabe und benutzt statt dessen die Fähigkeit des C16 und Plus4, Bildschirmfenster zu definieren.
Die folgenden Strings zur Druckersteuerung beziehen sich auf den Citizen 120-D im Epson-Modus, müßten also mit den meisten Druckern kompatibel sein. Wenn nicht, können sie durch die beigefügten Angaben leicht angepaßt werden. Ist das Programm geladen und gestartet, dann läuft nach kurzer Wartezeit die Eröffnungsgrafik.
Ein Tastendruck bringt uns in das Menü mit seinen vier Punkten. Der erste Punkt, Wochentag, dürfte klar sein. Zu jedem beliebigen Datum der christlichen Zeitrechnung bis zum Jahr 2299 rechnet das Programm den richtigen Wochentag aus.
Der zweite Punkt, Tagesdatum, rechnet für einen gegebenen Wochentag, Monat und Jahr die möglichen Daten aus. Ein Beispiel: Ihre Oma schwärmt immer davon, wie der Opa und sie sich am ersten Freitag im Mai 1948 kennengelernt haben. Benutzen Sie Menüpunkt 2 und schenken Sie der Oma am 7. Mai Blümchen. Der dritte Punkt ist im Ergebnis ähnlich, nur gibt er nach Eingabe von Wochentag, Tagesdatum und Jahr die möglichen Monate an. Beispiel: Einen Freitag, den 13., gab es im Jahr 1189 im Januar und Oktober.
Zum vierten Punkt, Festtage: Dieses Unterprogramm gibt für ein beliebiges Jahr die gesetzlichen Feiertage der Bundesrepublik Deutschland mit Datum und Wochentag an. Zusätzlich können vom Benutzer bis zu 16 weitere Daten eingegeben werden, zum Beispiel Geburtstage. Diese werden mit den gesetzlichen Feiertagen zusammen berechnet, und ein nach Daten sortierter Festkalender des Jahres wird wahlweise auf dem Bildschirm oder auf einem Drucker ausgegeben. Die Zusatzdaten werden programmintern gespeichert, so daß sie nicht für jedes neu abgefragte Jahr gesondert eingegeben werden müssen.
Das Programm rechnet nach der oben beschriebenen mittelalterlichen Kalenderrechnung. Nach Eingabe wird zunächst jedes Datum geprüft (kein 54. April möglich), dann wird der Sonntagsbuchstabe errechnet, anschließend je nach Unterprogramm Wochentag oder Tagesdatum.
Die Osterrechnung erfolgt der einheitlichen Handhabung wegen ebenfalls nach dieser alten Methode der Goldenen Zahl und der Ostergrenzen. Wer möchte, kann sich einmal an einem Programm für die Gauss'sche Osterrechnung versuchen; sie erfordert nicht soviele Data, hat aber nicht den altertümlichen Reiz der Sonnen- und Mondzyklen.
Zur genaueren Information über die Kalenderrechnung können folgende Bücher empfohlen werden:
Joachim Ekrutt: *Der Kalender im Wandel der Zeiten,* Stuttgart: Franckh'sche Verlagshandlung, 1972, Kosmos Bibliothek Bd. 274.
Für Leute, die recht genau über mittelalterliche Daten Bescheid wissen wollen:
Hermann Grotefend: *Taschenbuch der Zeitrechnung des deutschen Mittelalters und der Neuzeit,* Hannover: Hahnsche Buchhandlung, 12. Auflage, 1982.

fwk□

MASCHINENSPRACHE

Black Beauty: In den „Kleinen" steckt mehr, als Commodore eingebaut hat

Mehr als eine bloße Übersicht über Befehlssatz, Adressierungsarten und Flagbeeinflussungen bietet diese Einführung. Im Vordergrund stehen Übung und Praxis. Dafür sorgt das Programm CPU-TRAINER, mit dem CPU-Befehle direkt eingegeben und auf ihre Wirkung beobachtet werden können.

Viele Probleme lassen sich in BASIC nur unzureichend lösen. Oft ist BASIC zu langsam oder total ungeeignet. Der direkte Weg, der Griff zur Maschinensprache, ist vielen noch versperrt, da die nötigen Informationen fehlen. Sollte dem einen oder anderen der Befehlssatz der CPU-7501 oder 6502 bereits bekannt sein, so hilft dieses herzlich wenig, wenn er das Betriebssystem des Rechners nicht kennt.
Ohne die Kenntnis wichtiger I/O-Adressen und -Routinen läßt sich nicht einmal ein einziges Zeichen von der Tastatur einlesen oder auf den Bildschirm bringen. Wir besprechen daher auch die wichtigsten Kernel-Routinen, das Port-Register und einige interessante Register des TED-Chips, der für Bild und Ton zuständig ist.
Sie lernen, wie ein Ein- oder Ausgabekanal definiert werden kann, wie Sie Zeichen auf den Drucker senden, wie Sie etwas auf Kassette oder Diskette ausgeben können, und wie Sie sie auch wieder von dort einlesen. Wenn die CPU auch keine Multiplikation kennt, mit einem bestimmten Algorithmus läßt sie sich verwirklichen. Mit dem TEDMON, dem eingebauten Maschinenmonitor, können wir kleine Programme schreiben. Vom CPU-TRAINER aus können wir sie aufrufen und deren Ergebnisse kontrollieren.
Wie Sie sehen, ist der Rahmen unserer Einführung sehr weit gespannt. Wenn Sie diese gewissenhaft durcharbeiten, sind Sie bereits in der Lage, selbst kleine Maschinenprogramme zu verfassen. In den folgenden C16-P4-SPECIAL-Heften werden wir Ihnen, da Sie jetzt die Grundlagen dafür bekommen, erstaunliche Tips und Tricks zeigen, mit denen Sie das Betriebssystem des Rechners nach Ihrem Gutdünken manipulieren können. Ein Druckerpuffer, eine interruptgesteuerte Uhr oder ein nichtblinkender Cursor sind dann nicht mehr schwer zu programmieren.

I. DER CPU-TRAINER

Dieses Programm wird uns durch unseren ganzen Einführungskurs begleiten. Es ermöglicht die Direkteingabe von CPU-Befehlen.
Das Programm besitzt einen Maschinensprache- und einen BASIC-Teil. In *Zeile 100* wird das BASIC-Ende auf die Adresse $3000 herabgesetzt. Ab dieser Adresse können wir uns im Hauptspeicher unseres Rechners austoben. Der in den *DATA-Zeilen 130 bis 320* abgelegte Maschinenteil wird durch die *Zeilen 110 und 120* im Bereich von $065E bis $06D2 abgelegt. Er enthält am Anfang die Routine zur Übernahme der Registerinhalte in die CPU, zur Ausführung des CPU-Befehles und zum Zurückschreiben der CPU-Register in die dafür vorgesehenen Speicherstellen.
Die restlichen Routinen sorgen für die Sonderbehandlung der in *Zeile 1260* vermerkten Befehle. In *Zeile 330* werden die für die Registerinhalte reservierten Speicherstellen mit den Inhalten der CPU-Register oder mit dem Wert 127 initialisiert.
Ab *Zeile 370* beginnt der eigentliche BASIC-Teil. Es werden Variablen, die für die Bildschirmausgabe wichtig sind, die entsprechenden Inhalte zugewiesen. Interessant sind hier die *Zeile 480 und 490,* durch die die Inhalte der Variablen A$(0) bis A$(6) in *Zeile 380 bis 440* umgewandelt werden.
Die auf jedem Drucker darstellbaren Zeichen 0 und a bis j werden durch die in den *DATA-Zeilen 450 und 460* vermerkten Codes ersetzt, die nachher auf dem Bildschirm erscheinen sollen.
Der Bildschirmaufbau – *Zeile 570 bis 630* – begnügt sich mit wenig Programmzeilen, da geschickt auf die Bildschirmvariablen und Unterprogramme zurückgegriffen wird. Die Eingabe in den *Zeilen 680 bis 760* wurde nicht durch einen INPUT-Befehl realisiert.
Auffallend ist, daß auch kein GET- oder PRINT-Befehl im Programm vorkommt. Anstelle der Befehle, die auf ein definiertes Ein- oder Ausgabegerät zugreifen, oder dieses umdefinieren, verwendeten wir neben dem CHAR-Befehl die SYS-Aufrufe SYS60381 und SYS56393. SYS 60381 ersetzt den GET-Befehl. Das Zeichen wird unabhängig vom aktuellen Eingabegerät immer von der Tastatur eingelesen. Das abgefragte Zeichen wird von der Systemroutine im Akku abgelegt.
Nach dem SYS-Aufruf befindet es sich daher in Speicherstelle 2034, aus der wir es mit einem PEEK hervorholen können. Statt PRINT verwendeten wir SYS56393. Es kann nur ein einziges Zeichen ausgegeben werden. Dessen ASCII-Wert ist vorher mit einem POKE in die Speicherstelle 2034 zu bringen.
Der SYS-Aufruf lädt den Akku mit diesem Wert, die Systemroutine gibt das Zeichen unabhängig vom aktuellen Eingabegerät immer auf den Bildschirm aus. So wurde dafür Sorge getragen, daß auch bei umdefinierten Ein- und Ausgabegeräten unsere Bildschirmausgaben immer auf den Bildschirm gelangen und auch bei umdefiniertem Eingabegerät die Abfrage unserer Eingabe funktioniert. Nach einer Eingabe definieren wir uns in den *Zeilen 830 und 840* ein leeres Fenster. Bildschirmausgaben, die wir durch ein Maschinenprogramm veranlassen, sollen die Darstellung der CPU-Register nicht beeinflussen. Beim Sprung in den Monitor mit BRK soll uns der ganze Bildschirm zur Verfügung stehen *(Zeile 820).*

Nach der Befehlsabarbeitung sorgte *Zeile 680* wieder für den neuen Bildschirmaufbau. In *Zeile 850* wird durch den GOSUB-Befehl die eigentliche Abarbeitung veranlaßt. In *Zeile 1180* findet eine Verzweigung statt, wenn es sich nicht um einen Ein-Byte-Befehl handelt. Die Unterroutine in *Zeile 1050* untersucht, ob ein existierender Befehl vorliegt, und übergibt in diesem Falle einen weiter zu verarbeitenden Wert. Dieses ist im Normalfall der Operationscode für die CPU oder bei der Sonderbehandlung in *Zeile 1240 bis 1260* die Adresse der anzuspringenden Routine. In *Zeile 1200* wird der Operationscode an die richtige Stelle in das Maschinenroutine eingefügt, und nachher die Routine mit SYS1630 aufgerufen. Da für den Operationscode drei Byte reserviert sind, werden die verbleibenden zwei Stellen mit dem NOP-Code gefüllt, der die CPU zu keiner Operation veranlaßt.
In *Zeile 1380* bei den Zwei-Byte-Befehlen folgt dem Operationscode ein Datenparameter, daher ist hier nur noch ein NOP-Code erforderlich. In den *Zeilen 1550 bis 1570* bei den Drei-Byte-Codes folgen dem Operationscode zwei Adreß-Byte.

II. DIE ERSTEN SCHRITTE

Die Datenübergabe vom Speicher zu den Registern, von den Registern zum Speicher, dem Datentransfer zwischen den Registern, und das Retten von Registerinhalten auf den Stapel bekommen wir mit einigen wenigen Befehlen in den Griff.
Fünf der sechs CPU-Register sind auf dem Bildschirm dargestellt. Die Inhalte können wir in binärer, hexadezimaler und dezimaler Form betrachten. Ein Fragezeichen signalisiert, daß das Programm offensichtlich auf eine Eingabe wartet. Wenn wir jetzt CPU-Befehle in mnemonischer Form eingeben, können wir beobachten, wie die CPU-Register auf unsere Befehle reagieren. Machen wir uns also mit der Materie vertraut und starten unsere ersten Versuche.

Laden und Speichern

Wir geben ein:

```
LDA #$01
LDY #$02
LDY #$03
```

Wie wir wohl vermutet haben, zeigen unsere ersten drei Register die Werte eins, zwei und drei. Die Adressierungsart, die durch #$ gekennzeichnet wird, und die den darauffolgenden Wert in das Register lädt, heißt *unmittelbare Adressierung.*
Es existieren auch Befehle, die für sich allein stehen können, ohne daß noch irgendwelche Daten oder sonstige Parameter zu folgen brauchen. Diese Adressierungsart heißt *implizite Adressierung.* Ein sehr interessanter Befehl ist der Break, der mit BRK abgekürzt wird. Geben Sie doch einmal ein:

```
BRK
```

Das Bild auf dem Bildschirm hat sich auf einmal total verändert. Wir befinden uns jetzt nicht mehr in unserem BASIC-Programm, sondern im Maschinen-Monitor TEDMON, der in unseren Rechner von Haus aus eingebaut ist. Der BRK-Befehl löst einen Software-Interrupt aus, der beim C16/116/Plus4 und auch beim C128 zum Aufruf des TEDMON führt. Daß nichts mehr von unserem früheren Bildschirminhalt zu sehen ist, dafür kann der TEDMON nichts. Das CPU-Programm war so frei, für einen sauberen Bildschirm zu sorgen. Auch der Maschinen-Monitor zeigt uns die Registerinhalte, wenn auch nur in hexadezimaler Form allein.

SPRUNG IN DEN MONITOR

Ein Unterschied fällt sofort ins Auge. Gleich an erster Stelle findet sich der Programmzähler, der mit PC für *Programm Counter* abgekürzt ist. Dieses Register ist nicht nur ein Byte, sondern zwei Byte breit. Schließlich muß es alle Adressen von $0000 bis $FFFF erreichen können. Zur Zeit zeigt es auf eine Stelle im Maschinenteil unseres Programmes, wo es anschließend weitergeht, wenn wir mit einem Go dafür sorgen. Wir geben ein:

```
G
```

Nach dem Druck der RETURN-Taste zeigt der Bildschirm wieder das vertraute Bild mit den CPU-Registern. Sehr kurz war der Blick in den TEDMON. Vielleicht ist Ihnen bereits noch eine zweite kleine Abweichung aufgefallen. Prägen Sie sich doch einmal den Inhalt des Stapelzeigers ein und gehen dann wieder mit BRK in den Monitor.
Der Stackpointer SP weist einen ganz anderen Wert auf. Das CPU-Programm simuliert einen Stapelzeiger mit anderem Wert, damit beim Herumprobieren ein Programmabsturz nicht so leicht verursacht werden kann.
Ein sehr wichtiger Bestandteil des Rechners neben der CPU ist der Speicher. Mit dem Monitor können Sie sehr leicht Speicherbereiche betrachten. Wir geben ein:

```
M3000
```

Von der Adresse $3000 ausgehend, wird ein Speicherabschnitt in hexadezimaler Form dargestellt. Rechts finden wir invers die ASCII-Darstellung davon. Da ein Rechner auch Texte speichern soll, ist es interessant, zu sehen, wo ein Text abgelegt ist.
Das BASIC-Ende wurde vom CPU-Programm auf $3000 gelegt. Daher steht uns ab dieser Adresse der Hauptspeicher des Rechners fast uneingeschränkt zur Verfügung. Wir sorgen zunächst einmal für einen sauberen Speicher mit

```
F300,3FFF,0
```

Der Bereich von $3000 bis $3FFF ist nun vollständig mit Nullen gefüllt. Wir sehen uns nun wieder den ersten Abschnitt an $3000 mit

```
M3000
```

Wenn nun nach den folgenden Manipulationen andere Werte hier drinstehen sollten, kann keiner mehr sagen, dies wäre schon vorher der Fall gewesen. Wir können nun mit *G* getrost den Monitor wieder verlassen. Mit LDA, LDX und LDY nur unmittelbar Werte in die Register zu schieben, ist uns zu wenig, denn die Daten, um die es geht, befinden sich in der Regel irgendwo im Hauptspeicher des Rechners. Daten können nur durch die CPU verarbeitet werden. Daher muß es auch Befehle geben, die die Daten vom Speicher in CPU und wieder von der CPU in den Rechner befördern.
Beginnen wir mit dem Weg von der CPU in den Speicher. Uns stehen die Speicher- oder auch die Store-Befehle STA, STX und STY zur Verfügung. Diesen Speicherbefehlen muß eine Adreßangabe folgen, die angibt, wo im Hauptspeicher die Registerinhalte gespeichert werden sollen. Wir geben ein:

```
LDA #$54
STA $3000
LDA #$45
STA $3001
LDA #$53
STA $3002
LDA #$54
STA $3003
```

Diese Adressierungsart, bei der dem Operationscode die Speicheradresse

folgt, heißt *absolute Adressierung.* Gekennzeichnet wird sie durch das $-Zeichen, gefolgt von einer vierstelligen HEX-Zahl. Unsere CPU kennt auch noch weitere Adressierungsarten, bei denen nur ein einziges Byte in Form von zwei Hexziffern anzugeben ist, oder solche, bei denen das X- oder Y-Register mit zur Adreßbildung verwandt wird. Um das Programm nicht allzusehr aufzublasen, wurde darin auf weitere Adressierungsarten jedoch verzichtet.
Um den Erfolg unseres Speichertests zu begutachten, sollten Sie sich im Monitor nochmals den Bereich ab $3000 ansehen. Wie dieses vor sich geht, dürfte Ihnen inzwischen bereits bekannt sein. Wir finden die erwarteten Werte, die in ASCII-Darstellung das Wort TEST ergeben.
Nicht nur die Speicherbefehle, sondern auch die Ladebefehle lassen sich absolut adressieren. Spaßeshalber wechseln wir einmal das Register:

```
LDX $3000
STX $3008
LDX $3001
STX $3009
LDX $3002
STX $300A
LDX $3003
STX $300B
```

Wir haben damit die von $3000 bis $3003 abgelegten Werte nach $3008 bis $300B übertragen, wie uns die Überprüfung mit dem Monitor bestätigt.

Transferieren zwischen Registern

Datenübertragung von CPU-Register nach Speicher und umgekehrt kennen wir jetzt schon. Irgend etwas scheint aber noch zu fehlen. Wenn wir einen Wert verändern, so kommt es oft vor, daß wir gerne den ursprünglichen Zustand wiederherstellen würden. Im Akku könnte zum Beispiel ein Wert stehen, den wir für irgendwelche Zwecke benötigen. Vorher sollen aber noch ein paar Rechenoperationen erfolgen, die leider den Akkuinhalt verändern.
Eine Lösungsmöglichkeit ist mit unserem bisherigen Wissen bereits denkbar. So könnten wir den Akkuinhalt einfach an irgendeiner Adresse abspeichern und unseren Wert, sobald er wieder gebraucht wird, erneut einladen. Doch warum sollen Daten, die wir bereits in der CPU bereit haben, erst wieder in das RAM geschrieben und dann wieder hergeholt werden?
Speicherzugriffe kosten mehr Zeit als reine CPU-Operationen, und außerdem mehr Platz, da immer noch eine Speicheradresse anzugeben ist. Es sollte doch auch Möglichkeiten geben, Daten direkt von einem Register in das andere zu transferieren. Diese Transferbefehle gibt es selbstverständlich. Um Daten vom Akku in die Intexregister X und Y zu schaufeln, gibt es die Befehle:

```
TAX
TAY
```

Den umgekehrten Weg ermöglichen:

```
TXA
TYA
```

Scheuen Sie sich nicht, dies ruhig auszuprobieren. Solange nur ein oder zwei Werte gerettet zu werden brauchen, und die anderen Register nicht für andere Zwecke benötigt werden, ist dieses Verfahren ein brauchbarer Weg.
Oft werden aber Unterroutinen angesprungen, die alle Registerinhalte verändern. In diesen Unterroutinen kann wiederum der Bedarf bestehen, neue Registerinhalte zu retten. Als Ausweg erscheint in diesem Falle nur das Abspeichern in dafür vorgesehenen Speicherplätzen gangbar zu sein. Jede Unterroutine bräuchte also wieder ihre besonderen Speicherstellen zum Retten der Register.
Nicht leicht ist es wohl, da noch den Überblick zu behalten. Für rekursive Programmierung, bei der ein Unterprogramm sich selbst mehrmals aufruft, bringt auch dieses keine Lösung. Die vorgesehnen Speicherstellen würden bei jedem rekursiven Aufruf neu überschrieben werden, wodurch die früheren Werte verlorengehen. Das Stapelregister sorgt hier für Abhilfe.

Der Prozessorstapel

Laden Sie einmal einen beliebigen Wert in den Akku. Um diesen zu retten, geben wir jetzt den Befehl ein:

```
PHA
```

Wenn Sie dabei auf das Stapelregister geachtet haben, haben Sie sicherlich eine Veränderung wahrgenommen. Wenn nicht, so beobachten wir es bei einem erneuten Versuch. Laden Sie bitte den Akku mit einem neuen Wert und retten Sie diesen wieder mit PHA.
Der Wert des Stapelzeigers wurde jedesmal um den Betrag eins verringert. Wenn wir unsere geretteten Werte vom Stapel wieder abheben, so beobachten wir umgekehrt ein Erhöhen des Stapelzeigers. Vorher laden wir noch schnell den Akku mit einem anderen Wert, damit wir das Erscheinen der alten Werte sehen. Wir geben ein:

```
PLA
```

Es erscheint der zuletzt gerettete Wert. Nach einem erneuten PLA haben wir auch den zuerst geretteten Wert wieder. Der Stapelzeiger müßte nun wieder, wie am Anfang, den Wert 7F aufweisen. Um zu untersuchen, was geschehen ist, geben wir ein:

```
LDX $017F
LDY $017E
```

Wir sehen, vielleicht auch zu unserer Verblüffung, die ehemals geretteten Werte nun in den Indexregistern stehen.
Der Stapel funktioniert folgendermaßen: Für den Stapel ist ein RAM-Bereich von $0100 bis $01FF reserviert. Der Inhalt des Stapelzeigers gibt das Low-Byte der Adresse an, in die der Registerinhalt bei einem PUSH-Befehl geschrieben wird.
Der Stapelzeiger wird zusätzlich um den Wert eins erniedrigt, so daß er nun auf die nächstniedrige Adresse weist. So können verschiedene Daten hintereinander gerettet werden, ohne daß sie sich gegenseitig ins Gehege kommen.
Beim PULL-Befehl wird der Stapelzeiger wieder um Eins erhöht, und der unter der Adresse mit High-Byte 01 und dem Stapelzeigerinhalt als Low-Byte in das entsprechende Register geladen. Nicht nur der Akku kann „gestapelt" werden, sondern auch das Statusregister. Die Befehle hierfür sind:

```
PHP
PLP
```

Wir können somit alle Registerinhalte auf den Stapel retten:

```
PHP
PHA
TXA
PHA
TYA
PHA
```

Abheben müssen wir den Stapel in umgekehrter Reihenfolge:

```
PLA
TAY
PLA
TAX
PLA
PLP
```

Die Stapelbefehle erlauben uns gar, mit dem Statusregisterinhalt genauso, wie mit sonstigen Daten umzugehen. Wir bekommen die Werte in den Griff mit:

PHP
PLA

Das Zurückübertragen macht auch keine besonderen Schwierigkeiten:

PHA
PLP

Sogar das Stapelregister bekommen wir mit zwei Transferbefehlen völlig in den Griff:

TSX
TXS

Übersicht der besprochenen Befehle

Laden	LDA,	LDX,	LDY
Speichern	STA,	STX,	STY
Transferieren	TAX	TAY,	TSX
	TXA,	TYA,	TXS
Stapeln	PHA,	PHP	
	PLA,	PLP	

III. SCHALTEN MIT DER CPU

Einzelne Bit lassen sich mit den Befehlen ORA, AND und EOR setzen. Damit können wir verschiedene Umschaltungen in den TED-Registern vornehmen. Den Bildschirm auf 24 Zeilen oder auf 38 Spalten umzuschalten, ist relativ einfach zu handhaben.

Mit dem Computer lassen sich nicht nur Daten verwalten. Besondere Bausteine, die wir über bestimmte Adressen ansprechen können, erlauben uns, die verschiedensten Umschaltungen vorzunehmen. Oft ist es nur ein einziges Bit, das geändert werden soll. Die Änderung der anderen Bit könnte zu unerwünschten Ergebnissen führen.
Interessant sind vor allen Dingen die Multifunktionsregister des TED-Chips und der serielle Port unseres Rechners. Bevor wir uns mit diesen Bausteinen ausgiebig befassen, sind zuerst die Befehle zur Bit-Manipulation zu besprechen. Wir geben ein:

```
LDA #$00
LDA #$01
LDA #$02
LDA #$04
LDA #$08
LDA #$10
LDA #$20
LDA #$40
LDA #$80
```

Ein bestimmtes Bit läßt sich mit dem Ladebefehl leicht setzen. Der Nachteil ist, daß die bereits vorhandenen Bit leider auch verändert werden. Zum Verändern einzelner Bit eignet sich der Befehl EOR. Probieren Sie doch einmal:

```
LDA #$99
EOR #$01
EOR #$02
EOR #$04
EOR #$08
EOR #$10
EOR #$20
EOR #$40
EOR #$80
```

Wir haben ein Bit nach dem anderen umgedreht. Es lassen sich natürlich auch alle umdrehen mit:

```
EOR #$FF
```

Für uns könnte der EOR-Befehl bereits ausreichend sein, da wir ja sehen, ob ein bestimmtes Bit bereits den gewünschten Wert hat, oder nicht. Wenn nein, wenden wir den EOR-Befehl an, wenn ja, dann eben nicht. Die CPU nachsehen zu lassen, ob ein Bit gesetzt ist, oder nicht und gegebenenfalls mit dem EOR-Befehl zu reagieren, wäre wohl etwas aufwendig. Daher existieren die Befehle ORA und AND, mit denen unabhängig vom Bit-Zustand diesem in jedem Falle der Wert eins oder null zugewiesen werden kann:

```
LDA #$99
ORA #$01
ORA #$02
ORA #$04
ORA #$08
ORA #$10
ORA #$20
ORA #$40
ORA #$80
```

Ebenso wie beim EOR-Befehl lassen sich auch mehrere Bit setzen:

```
LDA #$81
ORA #$18
```

Die zwei mittleren Bit sind durch den ORA-Befehl hinzugekommen. Um zu wissen, welche Werte wir für den AND-Befehl brauchen, sollten wir die Werte, mit denen wir vorher geordert hatten, invertieren, und uns die Resultate merken:

```
LDA #$01
EOR #$FF
LDA #$02
EOR #$FF
LDA #$04
EOR #$FF
LDA #$08
EOR #$FF
LDA #$10
EOR #$FF
LDA #$20
EOR #$FF
LDA #$40
EOR #$FF
LDA #$80
EOR #$FF
```

Wir können nun den Akku wieder mit #$99 oder auch einem anderen Wert laden, und das Nullsetzen der Bit beobachten.

```
LDA #$99
AND #$FE
AND #$FD
AND #$FB
AND #$F7
AND #$EF
AND #$DF
AND #$BF
AND #$7F
```

Es lassen sich wieder mehrere Bit auf einmal setzen. Wir laden hierzu das X-Register.

```
LDX #$7E
```

Wir können sehen, daß die äußeren beiden Bit den Wert Null haben. Beim Undieren mit diesem Wert würden folglich die äußeren Bit auf Null gesetzt.

```
LDA #$99
AND #$7E
```

Probieren Sie ruhig noch ein wenig mit den Befehlen EOR, ORA und AND herum, denn Übung macht ja bekanntlich den Meister. Sie können auch die absolute Adressierung verwenden, bei welcher der Inhalt einer Speicherzelle mit dem Akkuinhalt verknüpft wird.

Die Register des TED-Chips

Die Register des TED-Chips werden angesprochen über die Adressen $FF00 bis $FF3F. Hier ist eine kleine Zusammenstellung:

$FF00	Low-Byte Timer A
$FF01	High-Byte Timer A
$FF02	Low-Byte Timer B
$FF03	High-Byte Timer B
$FF04	Low-Byte Timer C
$FF05	High-Byte Timer C
$FF06	Multi-Funktions-Register I
$FF07	Multi-Funktions-Register II
$FF08	Tastaturregister
$FF09	Interrupt Request Register
$FF0A	Interrupt Mask Register
$FF0B	Rasterinterrupt
$FF0C	Cursorpositon High-Byte
$FF0D	Cursorpositon Low-Byte
$FF0E	Tonhöhe Tongenerator 1
$FF0F	Tonhöhe Tongenerator 2
$FF10	Tonhöhe Tongenerator 2 Bit 8/9)
$FF11	Multi-Funktions-register III
$FF12	Multi-Funktions-register IV
$FF13	Zeichensatz-Adresse
$FF14	Basisadresse Farb-/Video-RAM
$FF15	Hintergrundfarbe
$FF16	Zeichenfarbe
$FF17	Multicolor-Farbe 1
$FF18	Multicolor-Farbe 2
$FF19	Rahmenfarbe
$FF1A	Bit-Mapping 1
$FF1B	Bit-Mapping 2
$FF1C	Aktuelle Rasterzeile Bit 8
$FF1D	Aktuelle Rasterzeile Bit 0–7
$FF1E	Rasterspalte
$FF1F	Vertikal-Adresse
$FF3E	Umschaltung auf ROM
$FF3F	Umschaltung auf RAM

Der TED-Chip besitzt eine ziemliche Anzahl von Registern. Viele davon sind erst bei fortgeschrittenen Systemkenntnissen interessant. Wir picken uns einige davon heraus, an denen wir sehr schön einige herausstechende Effekte beobachten können. Gut eignen sich hier die vier Multi-Funktions-Register, die Zeichensatzadresse, Hintergrundfarbe, Zeichenfarbe und Rahmenfarbe.
Zur Sicherung wird der Inhalt von $FF06 zusätzlich in das X-Register geladen.

```
LDA $FF06
TAX
```

Wir verändern die Bit null bis zwei, die angeben, um wieviel Pixel der Bildschirm vertikal verschoben werden soll.

```
ORA #$07
STA $FF06
AND #$F8
STA $FF06
STX $FF06
```

Beim Verschieben des Bildschirmes treten am oberen und unteren Rand unerwünschte Effekte auf. Bit drei, das den Bildschirm auf 24 Zeilen umschaltet, schafft vielleicht Abhilfe.

```
TXA
AND #$F7
STA $FF06
```

Nicht wie erwartet, verschwindet die letzte Zeile, sondern sowohl ein Teil der ersten und der letzten. Dieses müßte sich jedoch durch Bildschirmverschieben wohl wieder ausgleichen lassen.

```
ORA #$07
STA $FF06
AND #$F8
STA $FF06
```

Bei 24 Zeilen ist es möglich, den Bildschirm pixelweise ohne störende Nebeneffekte zu verschieben, bis schließlich die Breite einer ganzen Zeile erreicht wird. So können Smooth-Scrolling-Effekte programmiert werden. Wir stellen wieder den Anfangszustand her:

```
STX $FF06
TXA
```

Erschrecken Sie bitte beim nächsten Bit nicht, wenn plötzlich auf dem Bildschirm nichts mehr zu sehen sein sollte. Die Tastatur nimmt weiter Ihre Eingaben an.

```
EOR #$10
STA $FF06
STX $FF06
TXA
```

Sollen Programme etwas schneller laufen, oder Routinen völlig gleichmäßig ausgeführt werden, so ist es zweckmäßig, den Bildschirm auszuschalten. Die CPU wird so nicht mehr durch den TED-Chip gestoppt, damit dieser in der Zwischenzeit den Bildschirm aufbauen kann. Dies ist besonders für Signale wichtig, die nicht durch eine extra Clock-Leitung getaktet werden, sondern bei denen es einzig und allein auf die Impulsdauer ankommt. Bei der Datenübertragung zur oder von Datasette haben Sie diesen Effekt sicherlich schon beobachtet.
Einen Bildschirm, der nicht sehr viel herzeigt, bekommen Sie auch mit

```
EOR #$20
STA $FF06
STX $FF06
TXA
```

Multi-Funktions-Register I ($FF06)

Bit 0–2:	Vertikale Verschiebung des Bildschirms
Bit 3:	Umschaltung auf 24 Zeilen
Bit 4:	Bildschirm aus/ein
Bit 5:	Grafikmodus aus/ein
Bit 6:	Grafikzeichen werden als reverse Buchstaben wiedergegeben
Bit 7:	Funktion unbekannt

Multi-Funktions-Register II ($FF07)

Bit 0–2:	Horizontale Bildschirmverschiebung
Bit 3:	Umschaltung auf 38 Spalten. Schaltet unerwünschte Effekte beim Smooth-Scrolling aus.
Bit 4:	Multicolor enable
Bit 5:	Funktion unbekannt
Bit 6:	PAL/NTSC-Norm
Bit 7:	Invert off. Anstelle inverser Zeichen können Zeichen aus dem zweiten Zeichensatz mit dargestellt werden. Damit lassen sich gleichzeitig der Groß/Grafik- und der Klein/Groß-Zeichensatz darstellen.

Multi-Funktions-Register III ($FF11)

Bit 0–3:	Lautstärke (0–8)
Bit 4:	Tongenerator 1 ein/aus
Bit 5:	Tongenerator 2 ein/aus
Bit 6:	Tongenerator 2 Rauschen
Bit 7:	Tongenerator ein/aus

Multi-Funktions-Register IV ($FF12)

Bit 0/1:	Tonhöhe Tongenerator 1, Bit (8/9)
Bit 2:	Zeichensatz in RAM/ROM
Bit 3–7:	Basisadresse der hochauflösenden Grafik

Von Grafik ist leider keine Spur zu sehen, auch wenn vorher einmal der Grafikbildschirm aktiviert war, zeigt sich noch nichts von einer Grafik. Diese Bit-Umstellung hat nur eine Wirkung, wenn Sie zu ganz bestimmten Zeiten mit einer Vielzahl anderer Parameter zusammen erfolgt. Die Interruptroutine, die auch für die Bildschirmausgabe verantwortlich ist, macht beim Bearbeiten der Grafik von diesem Bit Gebrauch.
Fraglich ist, ob das nächste Bit uns sehr viel Nutzen bringen kann.

```
EOR #$40
STA $FF06
STX $FF06
TXA
```

Für irgendwelche schleierhaften Testzwecke soll angeblich Bit sieben dienen.

```
EOR #$80
STA $FF06
STX $FF06
TXA
```

Mit den Bit-Manipulationen dürften Sie nun schon ein wenig vertraut sein. So wird es wohl reichen, wenn wir ein paar weitere TED-Register ohne zusätzliche Experimentieranweisungen vorstellen.

Zeichensatzadresse ($FF13)

Die Bit zwei bis sieben enthalten die Adreß-Bit 10 bis 15 der Zeichengeneratoradresse. Wir starten einen Versuch.

```
LDX $FF13
```

Nun schalten wir mit Commodore-Taste und Shift den Zeichensatz um.

```
LDY $FF13
```

Der Wert der Zeichensatzadresse hat sich offensichtlich verändert. Wir schalten wieder um. Durch das Speichern der Werte in den Indexregistern ist auch ein Umschalten zwischen Groß- und Kleinschrift möglich.

```
STY $FF13
STX $FF13
```

Zeichenfarbe ($FF16)

Manipulationen dieses Registers bleiben ohne Wirkung. Die Zeichenfarbe bestimmt schließlich der im Farb-RAM abgelegte Code. Welcher Code abgelegt wird, bestimmt die Speicherstelle $053B. Wir laden diesen Wert.

```
LDA $053B
TAX
```

Bit null bis drei, die vier rechten Bit zeigen eine Null als Kennung für die Farbe schwarz. Bit vier bis sechs geben die Helligkeit an. Schwarz ist schwarz, deshalb ist die Helligkeit uninteressant. Die Frage ist, ob Bit sieben auch noch eine Funktion besitzt.

```
ORA #$80
STA $053B
STX $053B
TXA
```

Bit acht ist das sogenannte Blinkflag. Die neu ausgegebenen Registerwerte und unsere Eingabe blinkten fröhlich vor sich hin. Wenn Sie gerne Farb- und Helligkeitsänderungen vornehmen wollen, so dürfen Sie dieses gerne tun.

Rahmenfarbe ($FF19)

Mit diesem Register kann die Rahmenfarbe geändert werden. Dieses geht genauso wie bei der Zeichenfarbe. Bit sieben ist ohne Funktion. ein Blinken von Rahmen oder Hintergrund gibt es nicht.

Hintergrundfarbe ($FF15)

Es gilt das bereits in Verbindung mit der Rahmenfarbe Gesagte.

IV. SCHALTEN MIT DEM CPU-PORT ($0001)

Der C16/116/Plus4 besitzt als Prozessor keine CPU 6502, sondern die CPU 7501. Diese CPU zeichnet sich durch einen speziellen Port aus, den der Rechner zum Ansteuern von Kassettenport und seriellen Port benützt.

Wir sehen uns die Bit des Port-Registers an:

```
LDA $0001
TAX
```

Bit null bis drei, die vier rechten Bit, dienen zum Senden, Bit vier bis sieben, die vier linken Bit, zum Empfangen. Die Sendeleitungen laufen über einen Inverter. Wir sehen, daß Spannung auf den zu Bit null bis zwei dazugehörigen Leitungen liegt. Die zu Bit drei gehörige Leitung liegt auf Masse. Wir nehmen einige Manipulationen vor.

```
ORA #$01
STA $0001
LDA $0001
```

Der Zustand von Bit null hat sich geändert, allerdings ist mit Bit sieben das Gleiche passiert. Wenn wir Bit eins abändern, so können wir denselben Effekt auch bei Bit sechs beobachten. Bit sieben scheint irgendwie mit Bit null und Bit sechs mit Bit eins gekoppelt zu sein. Dieses ist auch der Fall. Mit Bit null schalten wir die DATA-Leitung, mit Bit eins die CLOCK-Leitung und mit Bit zwei die ATN-Leitung des seriellen Ports. Wer einen Spannungsmesser besitzt, kann dieses nachmessen.
Mit Bit sieben messen wir die Spannung auf der DATA-Leitung und mit Bit sechs die Spannung an der CLOCK-Leitung. Die ATN-Leitung können wir leider nicht abfragen. Hat Bit eins den Wert null, so liegt Spannung auf der DATA-Leitung, was Bit sechs mit einer Eins anzeigt. Im umgekehrten Fall zeigt Bit sechs keine Spannung an.
Wenn uns ein zweiter C16/116/Plus4 zur Verfügung steht, und wir ein serielles Kabel besitzen, können wir gar Signale von einem Rechner auf den anderen übertragen. Wir müssen hierzu im empfangenden Rechner das Sende-Bit auf Null setzen, so daß auf der entsprechenden Leitung fünf Volt anliegen. Wenn der Senderechner die Leitung des Sendebit auf eins setzt, fällt das Potential ab und der Empfangsrechner registriert dieses durch das Empfangs-Bit.
Serielle Übertragungen von Rechner zu Rechner sind so ohne Weiteres machbar. Bit eins dient nicht nur als Clock-Signal für den seriellen Port, sondern auch zur Ansteuerung der Write-Leitung des Kassettenports, was mit einem Voltmeter überprüft werden kann. Der Kassettenport stellt uns allerdings nur drei Volt zur Verfügung. Legen wir eine Spannung an die Read-Leitung des Kassettenports an, so sehen wir, daß Bit vier den Zustand dieser Leitung anzeigt.
Ob die Playtaste gedrückt wurde, oder nicht, darüber gibt uns das

Port-Register keinen Aufschluß, jedoch das Bit zwei in Speicherstelle $FD10. Ist dieses Null, so ist die Play-Taste gedrückt. Der Plus4 erlaubt uns über dieses Bit, den Kassettenmotor zu steuern. Da hier ein I/O-Baustein vorliegt, der auch für den Userport gedacht ist, können wir dort einfach etwas hineinschreiben und so eine gedrückte Rekordertaste simulieren.

LDA $FD10
EOR #$04
STA $FD10
LDA $0001

Der Plus4 zeigt im Port-Register, daß die Motor-Leitung, die durch Bit drei angesteuert wird, eingeschaltet ist. Das Spannungsmeßgerät sollte über sechs Volt anzeigen. Der C16/116 zeigt leider keinerlei Reaktion. Wir stellen beim Plus4 das Play-Flag wieder auf den ursprünglichen Wert

LDA $FD10
ORA #$04
STA $FD10
LDA $0001

Wir versuchen, die Motorleitung auf Spannung zu setzen.

EOR #$08
STA $0001
LDA $0001

Leider hatten wir keinen Erfolg. Wir schließen die Datasette an und drücken auf die Playtaste.

LDA $0001

Wie wir sehen, ist die Motorleitung nun angeschaltet. Wir können bei gedrückter Play-Taste nun nach Belieben schalten und walten.

EOR #$08
STA $0001
LDA $0001

Der Motor ist wieder aus.

EOR #$08
STA $0001
LDA $0001

Nun ist er wieder an. Für Schaltungen mit dem Kassettenport kann durch Verbinden der Play-Leitung mit Masse dafür gesorgt werden, daß wir auch die Motor-Leitung zum Schalten benutzen können.
Eine andere Möglichkeit gewinnen wir durch einen kleinen Speichereingriff:

LDY #$45
STY $0312

Damit ist die Betriebssystemroutine, die für die Kassettenmotoransteuerung bei gedrückter Play-Taste sorgt, ausgeschaltet. Auf einen Druck der Play-Taste allein hin rührt sich noch lange nichts. Der Motor wird nur noch durch Bit drei des Portregisters gesteuert. Auch wenn gar nichts angeschlossen ist, können wir dieses Bit jetzt nach Belieben verändern und die Spannung messen. Wir bringen nach unseren Versuchen unser Betriebssystem wieder in Ordnung.

LDY #$42
STY $0312

V. DIE LOGIK DER CPU

CPU-Operationen beeinflussen bestimmte Flags des Statusregisters. Acht bedingte Sprungbefehle reagieren auf die Zustände von vier Statusflags.

Viele sprechen von der Intelligenz des Computers. Alles, was eine CPU kann, ist jedoch nur auf ganz gewisse Schalterstellungen hin zu verzweigen. Damit diese Verzweigungen richtig stattfinden, ist die Intelligenz des Programmierers gefordert. Die Verzweigungsbefehle sind:

BEQ
BNE
BPL
BMI
BCS
BCC
BVS
BVC

Diese Befehle können Sie mit unserem CPU-Programm nicht eingeben. Sie sind nicht für sich, sondern nur in einem Programm brauchbar. Wir können uns jedoch mit den Schaltern befassen. Ein Register wurde bisher noch kaum besprochen, das Statusregister. Es ist sozusagen das Auge des Computers. Nur auf das, was das Statusregister anzeigt, kann die CPU mit Verzweigungen reagieren. Set- und Clear-Befehle erlauben uns die gezielte Einflußnahme auf einzelne Statusflags.

SEC
CLC

Das Carry-Flag ist das einzige Flag, das wir mit Set und Clear sowohl an- und ausschalten können und auf welches außerdem noch Verzweigungsbefehle reagieren, die Befehle BCS und BCC. Daher wird dieses Flag oft gesetzt, um den erfolgreichen Ablauf einer Operation zu signalisieren.
Zurückgekehrt aus einem Unterprogramm, das zum Beispiel Daten auf ein externes Gerät ausgeben sollte, läßt sich mit dem Branch-Befehl auf das Carry-Flag reagieren, für den Fall, daß die Ausgabe nicht geklappt hätte, da vielleicht der Drucker gar nicht angeschaltet war.

SEI
CLI

Mit diesen Befehlen können wir das Interrupt-Flag setzen und löschen. Wie wir wissen, kommt es zu Zwekken der Bildschirmausgabe und der Tastaturabfrage immer wieder zu Unterbrechungen des laufenden Programms. Dies kann aus irgendwelchen Gründen unerwünscht sein. Ist gar die RAM-Bank eingeschaltet und der Prozessor bearbeitet hierin ein Programm, so würde der Interrupt einen Absturz bewirken, wird doch durch diesen wieder die ROM-Bank aktiviert. Anstelle des Rücksprungs in die unterbrochene Routine, landet der Programmzähler dann irgendwo im Betriebssystem oder im BASIC-Interpreter, so daß dann nur mehr der Himmel weiß, was anschließend stattfindet.

SED
CLD

Damit eingegebene Zahlen nicht mühsam erst in Hex-Zahlen umgerechnet werden müssen, wenn wir Rechenoperationen vornehmen wollen, können wir dem Rechner durch Setzen des Dezimal-Flags mitteilen, daß er seine Additionen und Subtraktionen nun im sogenannten BCD-System vornehmen soll.

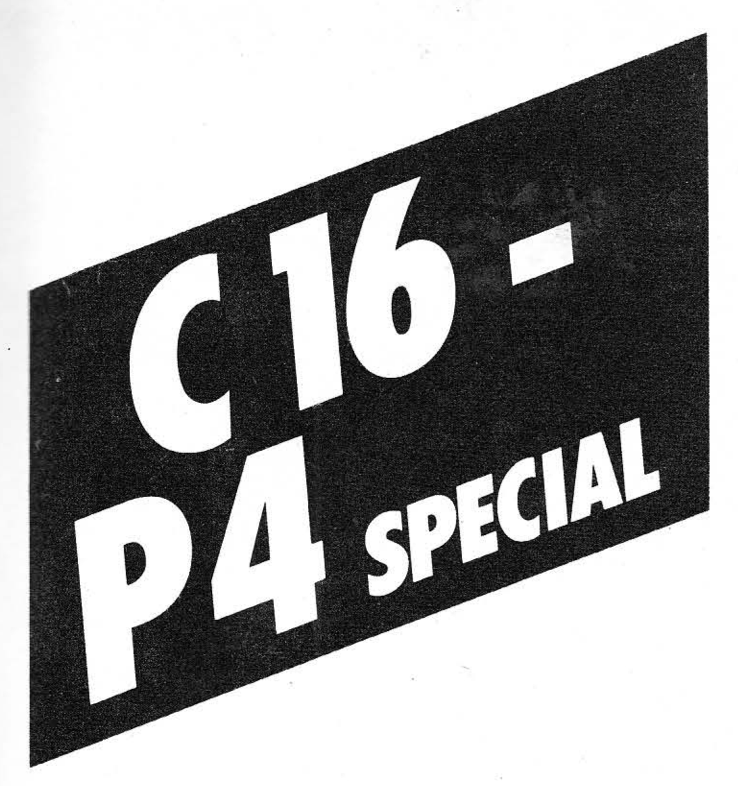
CLV

Bei Rechenoperationen mit vorzeichenbehafteten Zahlen ist das Überlauf-Flag von Interesse. Es sollte daher auch wieder zurückgesetzt werden können.

Zusammenfassung der Set- und Clear-Befehle

SEC
CLC
SEI
CLI
SED
CLI
CLV

Lesen Sie bitte weiter auf Seite 83



KOMMT REGELMÄSSIG ZU IHNEN INS HAUS

Finden Sie Ihr C16/P4 nicht am Kiosk? Weil es schon ausverkauft ist? Oder „Ihr" Kiosk nicht beliefert wurde? Kein Problem! Für ganze 70 DM liefern wir Ihnen per Post sechs Hefte ins Haus (Ausland 80 DM). Einfach den Bestellschein ausschneiden – fotokopieren oder abschreiben, in einen Briefumschlag und ab per Post (Achtung: Porto nicht vergessen). C16/P4-SPECIAL kommt dann pünktlich ins Haus.

WICHTIGE RECHTLICHE GARANTIE!

Sie können diesen Abo-Auftrag binnen einer Woche nach Eingang der Abo-Bestätigung durch den Verlag widerrufen – Postkarte genügt. Zur Wahrung der Frist genügt die rechtzeitige Absendung. Ansonsten läuft dieser Auftrag jeweils für sechs Ausgaben, wenn ihm nicht vier Wochen vor Ablauf widersprochen wird, weiter.

DAS SONDERANGEBOT: PRIVATE KLEINANZEIGEN KOSTENLOS!

Das bietet Ihnen COMMODORE-WELT: KLEINANZEIGEN SIND KOSTENLOSE FÜR PRIVATANBIETER! Suchen Sie etwas, haben Sie etwas zu verkaufen, zu tauschen, wollen Sie einen Club gründen? Coupon ausfüllen, auf Postkarte kleben oder in Briefumschlag stecken und abschicken. So einfach geht das. Wollen Sie das Heft nicht zerschneiden, können Sie den Coupon auch fotokopieren. Oder einfach den Anzeigentext uns so schicken, auf Postkarte oder im Brief. Aber bitte mit Druckbuchstaben oder in Schreibmaschinenschrift!

Und: Einschließlich Ihrer Adresse und/oder Telefonnummer sollten acht Zeilen à 28 Anschläge nicht überschritten werden.

ACHTUNG: WICHTIGER HINWEIS!

Wir veröffentlichen nur Kleinanzeigen privater Inserenten kostenlos. Gewerbliche Anzeigen kosten pro Zeile 4,80 plus Mehrwertsteuer!

Wir versenden für Privat-Inserenten keine Beleg-Exemplare!

Chiffre-Anzeigen sind nicht gestattet! Wir behalten uns vor, Anzeigen, die gegen rechtliche, sittliche oder sonstige Gebote verstoßen, abzulehnen!

Anzeigenabdruck in der Reihenfolge ihres Eingangs, kein Rechtsanspruch auf den Abdruck in der nächsten Ausgabe!

Die Insertion ist nicht vom Kauf des Heftes abhängig! Wir behalten uns vor, Anzeigen, die nicht zum Themenkreis des Heftes – Computer – gehören, nicht abzudrucken oder sie nur insoweit zu berücksichtigen, wie es der Umfang des kostenlosen Anzeigenteils zuläßt.

ANZEIGENSERVICE

Die große Börse für jeden Zweck in der C16/P4-SPECIAL. Kostenlos für Privat-Inserenten. Spottbillig für gewerbliche Anbieter. Einfach Coupon ausschneiden, fotokopieren o.ä., ausfüllen und ab die Post – Freimachen nicht vergessen! – Unsere Adresse steht auf dem Coupon, ebenso die Preise für gewerbliche Anbieter! Achtung! Wir weisen ausdrücklich darauf hin, daß wir offensichtlich gewerbliche Anzeigen nicht kostenlos veröffentlichen und uns jedweden Abdruck kostenloser Anzeigen vorbehalten müssen, insbesondere, wenn deren Inhalt nicht CBM-typisch ist oder gegen geltendes Recht verstößt. Private Chiffreanzeigen werden nicht aufgenommen. Für gewerbliche Anbieter DM 4,80 pro Zeile zuzüglich gesetzlicher Mehrwertsteuer.

COMMODORE WELT
ANZEIGENABTLG
POSTFACH 1161
D-8044 UNTERSCHLEISSHEIM

Name ____________

Vorname ____________

Straße/Hausnr. ____________

Plz/Ort ____________

ABO-SERVICE-KARTE

16/III **COUPON**

Ich nehme zur Kenntnis, daß die Belieferung erst beginnt, wenn die Abo-Gebühr dem Verlag zugegangen ist.

Ja, ich möchte von Ihrem Angebot Gebrauch machen. Bitte senden Sie mir bis auf Widerruf ab sofort jeweils die nächsten sechs Ausgaben an untenstehende Anschrift. Wenn ich nicht vier Wochen vor Ablauf kündige, läuft diese Abmachung automatisch weiter.

Name ____________ **16/III**

Vorname ____________

Straße/Hausnr. ____________

Plz/Ort ____________

Ich bezahle:

- ☐ per beiliegendem Verrechnungsscheck
- ☐ gegen Rechnung
- ☐ bargeldlos per Bankeinzug von meinem Konto

bei (Bank) und Ort ____________

Kontonummer ____________

Bankleitzahl ____________
(steht auf jedem Kontoauszug)

Unterschrift ____________

Von meinem Widerspruchsrecht habe ich Kenntnis genommen.

Unterschrift ____________

COMMODORE WELT
ABO-SERVICE **16/III**
POSTFACH 1161
D-8044
UNTERSCHLEISSHEIM

LOAD&RUN



Das Spiele-Magazin

Hallo, liebe Leser,

Infogrames scheint umgestellt zu haben. Spiele, wie wir sie bisher von der Firma gewohnt waren, werden kaum noch produziert. Dafür hat man sich jetzt auf die Umsetzung von erfolgreichen Comic-Bänden bekannter Autoren spezialisiert.
Nach der gelungenen Bob-Moran-Trilogie, die wir für LOAD & RUN 5/88 getestet haben, erscheint nun ein neues Programm auf dem deutschen Softwaremarkt. Jeder kennt Bobo, den Ausbrecherkönig der Haftanstalt Riegelfest, der bereits seit 17 Jahren unsere Lachmuskeln strapaziert. Zeichner Paul Deliege war mit seiner Comicfigur damals der Durchbruch gelungen. Anfangs fanden Bobos Abenteuer lediglich in Rolf Kaukas Fix und Foxi Platz. Mittlerweile gibt es viele Sonderbände mit Bobo und seinen Kumpanen vom Zuchthaus Riegelfest.
Infogrames ist mit seinem neuesten Werk Bobo ein Meisterstück gelungen. Wir sind gespannt, wann das Spiel in den internationalen Software-Hitlisten auftauchen wird. Einen ausführlichen Testbericht lesen Sie in dieser Ausgabe.

Die Players Pages sind ja bereits ein fester Bestandteil von LOAD & RUN. Da diese Spielehilfe-Seiten bei unseren Lesern großen Anklang gefunden haben, wie die vielen Zuschriften beweisen, werden wir sie auch in Zukunft beibehalten. Auch in diesem Monat kommen die Schneider/Amstrad-CPC-Besitzer sehr gut weg. Eine lange Poke-Liste für den CPC soll den bösen Sprites wieder einmal das Leben schwer machen.
An dieser Stelle möchte ich Sie nochmals aufrufen, sich an der Gestaltung der Players Pages aktiv zu beteiligen. Auf gut deutsch: Schicken Sie uns Pokes, Karten, Tips, Tricks, Lösungen oder Fragen zu jedem x-beliebigen Spiel für jedes x-beliebige Computersystem. Als kleinen Ansporn verlosen wir jeden Monat ein Originalspiel, deshalb Systemangabe nicht vergessen! Ich hoffe auf große Resonanz von Ihrer Seite.

In diesem Sinne wünsche ich Ihnen im Namen der Redaktion viel Spaß mit dem neuen LOAD & RUN.

Ihr
Thomas Bosch

Inhalt

Schmetterling durch CPC
Volleyball-Simulator für den Sportspielefreak
Seite III
Pink Panther
Der Rosarote Panther, nach Kino und Fernsehen auch als Geschicklichkeitsspiel
Seite IV
Bobo
Der König der Ausbrecher, in seinem Element, dem Gefängnis Riegelfest
Seite V
Vier auf einen Streich
Deep Space – Hacker II – Brataccs – LCP
Seite VI
Troll
Ein Jump- and Run-Game für geschickte Spieler
Seite VII
Top Fuel Eliminator
Ein neues Autorennspiel, das wirklich „top" ist
Seite VIII
Arkanoid II
Die Breakout-Welle im erwarteten Aufschwung
Seite IX
Rimrunner
Fleißiges Ballern ist bei diesem Game angesagt
Seite X
10 Great Games II
Software-Sampler für den Commodore 64, für jeden Geschmack etwas
Seite XI
Games für kleine Geldbeutel
On Cue – The Official FA Cup – Impossible Mission – Bruce Lee – The Eidlon
Seite XII
Kurzberichte
Trauma – Bob Moran – Witzball – Iznogud – Patton vs. Rommel – Starfleet
Seite XIV
Computer Classics
Dynamite Dan – Zynaps – Into The Eagle Nest – Aliens – The Pumpkin Strikes Back
Seite XV
Players' Pages
Tips & Tricks für aktive Joystickartisten **Seite XVI**



Auf dem Softwaremarkt gibt es die unterschiedlichsten Sportsimulationen. Vom Fußball über Handball bis zum Eishockey kann sich der Sportfreak an seinem Computer zumindest theoretisch austoben. Eine Kategorie fehlte bisher: Volleyball.

Uns stand für den Test die Kassettenversion für den Schneider CPC zur Verfügung. Die etwas ungewöhnlich lange Ladezeit wird durch mehrere Bilder und Grafik-Effekte überbrückt. Danach steht ein Menü mit vier Optionen zur Verfügung, welche mit den Cursor-Tasten angewählt werden. Hier können zunächst einige Voreinstellungen wie Verändern der Bildschirmfarben (wichtig für Grünseher!) oder Auswahl der Steuerung tätigen. Auch ein abgespeicherter Spielstand kann geladen werden.

Bevor Sie die Option Play anwählen, müssen Sie sich entscheiden, welche Rolle der Computer im Spiel übernehmen soll.

Bis zu 14 Spieler, also zwölf Feldspieler und zwei Trainer, können am Volleyball-Turnier teilnehmen. Die größere Zahl der Spieler erhöht den Spaß.

Für Einzelspieler gibt es mehrere Möglichkeiten. Der Computer kann die volle Kontrolle über die Mannschaft übernehmen. In diesem Fall fungiert der Spieler lediglich als Trainer und muß im geeigneten Moment die Taktik wechseln.

Eigene Taktiken können mit dem integrierten Tatik-Editor entworfen werden. Der Spieler kann seine Mannschaft selbst steuern.

Sehr gut gelungen ist dem Autor die Animation der Spielfiguren und des Balles, welche sich zwar etwas langsam, dafür aber völlig flimmerfrei über den Bildschirm bewegen. Untermalt wird das ganze Spektakel von diversen Sound-Effekten wie dem Aufklatschen des Balles oder dem Pfeifen des Schiedsrichters.

Durch zahllose Varianten, Möglichkeiten und Optionen wird mit dem Volleyball-Simulator jedem die Möglichkeit gegeben, „sein" Spiel zu machen, egal, ob jemand schon mal einen Ball in der Hand gehalten hat oder nicht. Die 50seitige deutsche Anleitung führt ausführlich und informativ in das Spielgeschehen ein, so daß auch Volleyball-Neulinge bereits nach wenigen Seiten die wichtigsten Regeln beherrschen. ● *TB*

Schneider Kass. deutsch/engl.

Titel:	The Volleyball-Simulator
Getestet:	Schneider CPC
Umsetzungen:	C-64, C-128 (geplant: Amiga, Atari ST)

Im Test:	Preis (DM):	
☐ Diskette		☒ Joystick
☒ Kassette	29,95	☒ Tastatur
		☐ Maus

Wertung	0	25	50	75	100
Grafik					
Sound					
Bedienung					
Motivation					

Pink Panther

Nach ihrem Superhit Vampire's Empire brachte das Edel-Softwarehaus Magic Bytes ein qualitativ kaum unterlegeneres Produkt auf den deutschen Markt. Das Geschicklichkeitsspiel „Pink Panther" bringt die Handlung der bekannten Fernsehserie in Ihren Rechner.



Die Handlung von Pink Panther kann nicht gerade als ehrenwert bezeichnet werden. Unser Held Paulchen ist mal wieder total abgebrannt und braucht dringend Geld, um sich über Wasser zu halten. Deshalb sucht er sich einen Job bei den Reichsten der Stadt als Hausdiener. Allerdings hat er dabei einen ganz gemeinen Hintergedanken: Tagsüber in den noblen Häusern wohnen und nachts dann eben diese von sämtlichen Wertgegenständen befreien.

Reiche Hausherren ausrauben

Fünf Hausherren würden schon ausreichen, um sich einen Urlaub im sonnigen Süden leisten zu können. Leider hat er dabei nicht damit ausgerechnet, daß die Eigentümer Schlafwandler sein könnten. Auch Inspektor Clouseau ist nicht untätig und versucht, Paulchen auf frischer Tat zu ertappen. Deshalb schnüffelt auch er nachts ständig durch die Häuser. Paulchen muß also aufpassen, daß er während seiner nächtlichen Raubzüge weder Clouseau trifft, noch daß ein Hausherr versehentlich gegen den Wandschrank schlafwandelt und dadurch unsanft bei seinem Spaziergang gestört wird. Gelingt es ihm, eine Villa komplett auszuräumen, kann er mit der nächsten weitermachen.

Insgesamt gibt es fünf Villen auszuräubern, bis das Spiel gewonnen ist.
Ist das Spiel geladen, so stehen Sie vor der Wahl, ob Sie ins Kaufhaus gehen, um sich passend zu kleiden, oder sich aber gleich um einen Job bewerben, was für den Anfang wohl das Vernünftigere ist. Nach einem kurzen Gespräch mit Ihrem neuen Herrn können Sie dann gleich mit der Arbeit anfangen. Natürlich kommt in diesem Game nur das nächtliche — wesentlich interessantere — Leben Paulchens zur Geltung. Sie sehen das Haus, dessen Etagen und Einrichtungen wie Möbel und Bilder von der Seite. Bewegen Sie den Joystick, so scrollen der Vorder- und Hintergrund fließend in alle Richtungen mit.

Die Grafiken aller Objekte sind hervorragend gezeichnet und stimmen in allen Details, auch in der Farbe, mit dem Trickfilm-Vorbild überein. Die Programmierer haben wirklich auf alles Rücksicht genommen. Sogar die Knöpfe an den Schränken und an Stiegengeländern sind hervorragend zu erkennen. Die Animation der Figuren läßt eigentlich nichts zu wünschen übrig. Sauber und fließend wandelt der schläfrige Hausherr durch die Flure, während ihm der Panther alle möglichen Gegenstände aus dem Weg räumt. Inspector Clouseau spaziert pixelgenau durch die Gegend.
Der Sound hingegen ist etwas zu kurz gekommen. Nur ein monotones Gedudel des berühmten Soundtracks wird rauf und runter gespielt. Daneben sind natürlich noch die üblichen Hintergrundgeräusche à la „Knacks" und „Rumms" zu hören. Wenigstens die Melodie hätten die Programmierer ansprechender machen können.
Auch mit der Motivation sieht es nicht ganz so gut aus. Viel Action und Hektik sind bei der ganzen Handlung nicht drin. Bereits nach einigen nächtlichen Streifzügen wird die Ausrauberei langweilig, da sich nichts Neues mehr bietet.

Fazit

Pink Panther kommt leider nur in Sachen Grafik an Vampire's Empire heran. In allen anderen Punkten, besonders aber im Spielspaß, schleicht Pink Panther weit hinter ihm her. Technisch ist das Game jedoch durch und durch gelungen. Freunde von ruhigeren Denk- und Geschicklichkeitsspielen sollten am rosaroten Panther nicht vorbeisehen. ● *mn*

Titel:	**Pink Panther**
Getestet:	**Amiga**
Umsetzungen:	**Atari ST, Schneider CPC, C-64**

Im Test: ☒ Diskette ☐ Kassette
Preis (DM): k.A.
☒ Joystick ☐ Tastatur ☐ Maus

Wertung (0 – 25 – 50 – 75 – 100)
Grafik
Sound
Bedienung
Motivation

Bobo

Jeder Comic-Fan kennt Bobo, den König der Ausbrecher. Seit nunmehr 17 Jahren versucht er, aus dem Gefängnis Riegelfest auszubrechen, weil er dort ständig schikaniert wird. Infogrames hat ihre „Collection Spirou" um die Abenteuer Bobos erweitert.



Sie werden für den Atari ST mit ROM-TOS auf zwei Disketten geliefert und bestehen aus sechs Einzelspielen, die Sie entweder der Reihe nach oder auch einzeln spielen können. Maximal sechs Spieler können sich an dem Spektakel beteiligen.

Das erste Spiel läuft in der Kantine des Gefängnisses ab. Bobo muß seinen hungrigen Kameraden Essen auftischen, sonst droht eine Meuterei. Schafft er es nicht, alle zufrieden zustellen, bekommt er den Napf aufgesetzt und das Spiel ist beendet.

In einer weiteren Szene muß Bobo einen großen Haufen Erdäpfel schälen. Gar nicht so einfach, da der Haufen ständig größer wird. Die Sache muß gründlich gemacht werden, denn schlecht geschälte Kartoffeln kommen wieder zurück. Dieses Spiel ist beendet, sobald Bobo von ungeschälten Kartoffeln überschwemmt wird.

Den Boden aufzuwischen ist Bobos verhaßteste Arbeit, denn seine Kumpane haben die schlechte Angewohnheit, genau da vorbeizugehen, wo unser Held den Putzlappen schwingt. Da der Boden feucht ist, hinterlassen sie Spuren und Bobo muß hinterherwischen, denn alles muß blitzblank sein, wenn der Herr Direktor zur Inspektion kommt.

Wenn die Gefängiswärter mal wieder streiken, versuchen natürlich die Insassen, auszubrechen. Diesmal springen sie vom Fenster herab auf ein Trampolin, das sie dann über die Mauer befördert. Bobo, intelligent wie er ist, hat die Aufgabe übernommen, das Trampolin von Fenster zu Fenster zu schieben. Verständlich, daß seine Kumpane nicht begeistert sind, wenn sie ins Leere springen.

Im nächsten Spiel ist Bobo endlich die Flucht gelungen. Er balanciert auf Stromkabeln herum und muß dabei von einem auf das andere springen, um den Spannungsblitzen auszuweichen, die seiner Gesundheit schaden können.

Nach einem erfolgreichen Tag legt sich ein professioneller Ausbrecher müde ins Bett und möchte schlafen. Leider schnarchen die Zellenkollegen so laut, daß dies unmöglich ist. Bobo bleibt nichts anderes übrig, als aufzustehen und jeden einzelnen zu beruhigen. Dabei darf er nicht über Nachttöpfe oder ähnliche Gegenstände stolpern.

Alle sechs Spiele sind grafisch sehr sauber und detailliert gestaltet. Die Bewegungen der Figuren erfolgen völlig fließend und ruckfrei. Zu jedem Spiel erklingt eine flotte Hintergrundmusik, die den Spieler gegeneinander anspornt. Jedes Spiel verfügt zusätzlich über einige nette Effekte. Es kann zum Beispiel passieren, daß beim Trampolinspringen ein Gefangener zu flach abspringt und, statt über die Mauer zu segeln, mit einem dumpfen „Boing" dagegen kracht, langsam runterrutscht, den Kopf schüttelt und erneut das Absprunggebäude betritt.



Fazit

Bobo ist ein Spiel, das einfach Spaß machen muß, vor allem dann, wenn mehrere Spieler gegeneinander antreten. Wettbewerbe wie beispielsweise ein Punktekampf können problemlos realisiert werden. Die gute Grafik und der gelungene dreistimmige Sound lassen die Spielmotivation enorm ansteigen. Wer Bobo noch nicht kennt, sollte sich sofort auf den Weg zum Computer-Fachhändler machen. ● *TB*

Titel: Bobo
Getestet: Atari ST
Umsetzungen: C-64, C-128

Im Test: ☒ Diskette ☐ Kassette

Preis (DM): 59,95

☒ Joystick
☒ Tastatur
☐ Maus

Wertung 0 25 50 75 100
Grafik
Sound
Bedienung
Motivation

Vier auf einen Streich

Eine recht günstige Anschaffungsgelegenheit für die ansonsten nicht billige Amiga-Software sind die mittlerweile beliebt gewordenen Game-Sampler. Mit ihnen erhält man mehrere Spiele, die zwar oft nicht mehr zu den Neuesten gehören, aber trotzdem gut sind, zum Preis von einem.

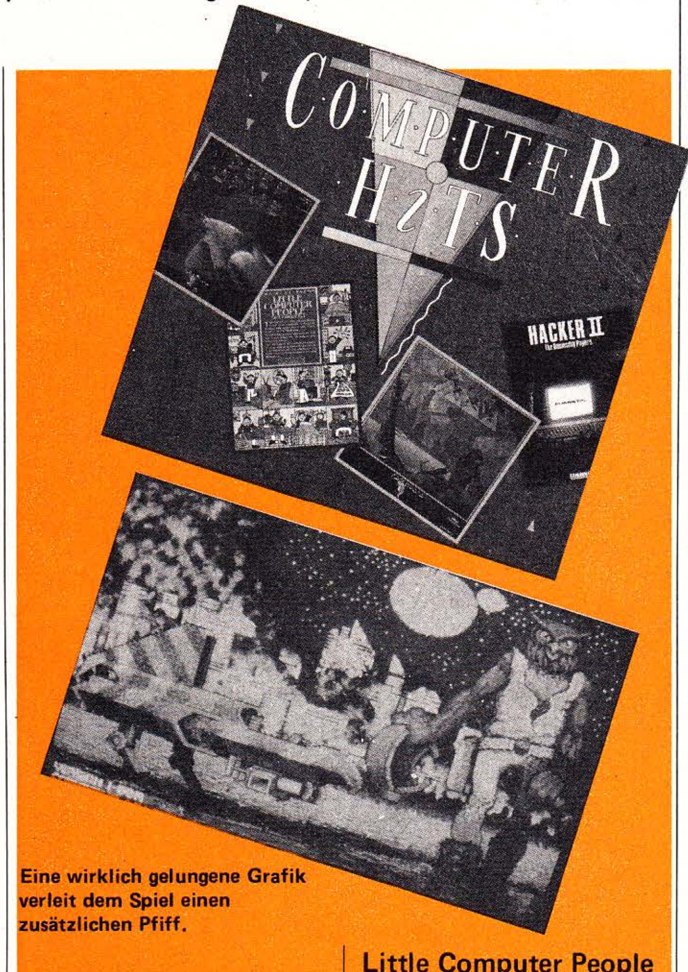
Ariolasoft vertreibt seit einiger Zeit in Deutschland einen neuen Sampler namens „Computer Hits".
Mit den vier Titeln handelt es sich um Deep Space, Hacker II, The Little Computer People und Brataccs.

Deep Space – Krawall im Weltraum

Bei Deep Space handelt es sich um ein Weltraum-Spiel mit dem Blickpunkt aus dem Cockpit eines Kampfraumschiffes. Wie bei vielen anderen Games dieser Sorte kommen alle erdenklichen Gegenstände und Objekte auf Sie zugeflogen. Darunter sind, wie sollte es auch anders sein, auch Asteroiden und feindliche Raumjäger, die Sie unbedingt abknallen sollten, da nämlich Sie sonst schnell vom Himmel gefegt werden. Das Spiel verfügt über einige Extrafunktionen, wie verstellbare Radarschirme und Schutzschilde, die recht hilfreich sein können. Im Großen und Ganzen ein ganz nettes Spiel für zwischendurch, das aber schnell eintönig wird.

Hacker II – The Doomsday Papers

Bei Hacker II sieht es da schon ganz anders aus. Hierbei handelt es sich um ein technisch sehr kompliziertes und forderndes Spiel, das Sie sicher länger vor dem Bildschirm festhalten wird.



Eine wirklich gelungene Grafik verleit dem Spiel einen zusätzlichen Pfiff.

Bei ihm müssen Sie per Datenfernübertragung einen Roboter in einem unterirdischen Bunkerkomplex in Sibirien umhersteuern und mit seiner Hilfe wichtige amerikanische Geheimdokumente finden. Die Wachen und Video-Überwachungskameras, die Sie übrigens manipulieren können, versuchen das natürlich zu vereiteln. Grafik und Sound liegen beim Durchschnitt.
Insgesamt ist Hacker II ein durchweg ansprechendes und interessantes Game.

Little Computer People

Wußten Sie schon, daß in Ihrem Rechner ein kleines Männchen wohnt? Und dieses Männchen wartet nur darauf, endlich den kennenzulernen, dessen Befehle es Tag für Tag millionenmal ausführt.
Sie werden sich wundern, wie bequem es sich unsere Little Computer People in der langweiligen IC-Landschaft gemacht hat. Ein schönes großes Haus mit allem drum und dran, inklusive Hund und Klavier, paßt leicht in ihren Computer. Mit Hilfe eines komplizierten, aus Forschungsgründen generierten Programm ist es gelungen, dieses Haus auf dem Monitor sichtbar zu machen.
Little Computer People kann eigentlich nur bedingt als Spiel bezeichnet werden, da es hier nichts zu spielen gibt – nur zu beobachten.

Unterhaltung per Tastatur

Über die Tastatur können Sie sich mit dem Männchen unterhalten und es wird dementsprechend darauf reagieren.
Die Grafik ist gelungen und sehr niedlich ausgefallen. Ein „Spiel", mit dem Sie sich länger beschäftigen können, da offenbar noch niemand seinen privaten Mann genau kennt. Überraschenderweise tauchen immer wieder völlig neue Situation auf. Mein Computermännchen hat sich zum Beispiel immer strikt geweigert, ein Bad zu nehmen. Nachdem ich ihn etwas aggressiver dazu aufgefordert und sogar geschimpft habe, ist er murrend aber zügig in die Wanne gestiegen.

Brataccs – Gejagt vom ganzen Universum

Das letzte Game dieses Samplers bereitet bei seiner Beurteilung gemischte Gefühle. Obwohl die Idee gut und das Spiel an sich sauber programmiert ist, kann es nicht recht überzeugen.
Sie sind ein Flüchtling, der von Polizei und Mafia gejagt wird. Ihnen wird das schwere Verbrechen des „genetischen Betrugs" zur Last gelegt. Was immer das auch sei, jedenfalls haben Sie beschlossen, von der Erde zu verschwinden und sich auf einen kleinen Bergwerksasteroiden, Brataccs, tele-

portiert. Hier soll irgendwo der Beweis Ihrer Unschuld sein. Natürlich wissen Sie nicht im geringsten, wie der Beweis aussieht und müssen sich somit mit der Bevölkerung Brataccs' auseinandersetzen. Wie bei allen Wesen gibt es auch hier die unterschiedlichsten Typen, manche freundlich, manche feindlich gesinnt.
Als Spieler führen Sie Ihren Schützling per Tastatur oder Maus über den Bildschirm. Die Steuerung ist derart komplex und empfindlich, daß sie dem Anfänger einige Schwierigkeiten bereitet.
Die Grafik erreicht nicht den heutigen Standard. Vom Sound wollen wir gar nicht reden. Das wäre alles nicht so schlimm. Aber woran es Brataccs wirklich fehlt, ist Spielwitz. Bereits nach einigen Minuten wird das Spiel langweilig. Für Freaks, die sich gerne mit komplizierten Games beschäftigen ist es aber sicher gar nicht so schlecht.

Fazit

Computer Hits enthält teilweise Spiele aus der Anfangszeit des Amiga, als er noch mindestens 4.000 Mark kostete. Deshalb nutzt keines der Games den Rechner vollkommen aus. Bedenkt man aber, daß Sie zum (anfangs etwas hoch erscheinenden) Preis von knapp achtzig Mark gleich vier Games bekommen, so wird sich mancher überlegen, ob er nicht zugreift. Zwei der Spiele sind unbedingt empfehlbar: Hacker II und Little Computer People. Die anderen haben einige Macken, gehören aber trotzdem nicht zur schlechtesten Sorte. ● *mn*

Troll

Jump- and Run-Spiele sind in. Um sie erfolgreich auf den Markt zu bringen, muß man sie nur neu verkleiden und sich immer wieder fantasiereiche Geschichten dazu einfallen lassen. Ein neues Produkt nach diesem Schema nennt sich kurz und bündig Troll und verdankt seine Existenz dem Hause Denton Designs.

Die Story: Die Hauptperson ist ein etwas zu groß geratener, aber dennoch recht liebenswerter Troll namens Humgruffin. Dieser ist durch einen Fluch in die Unterwelt von Narc geraten, ein unheimliches Land voller Kristallhöhlen, bevölkert von einem Stamm von Kobolden, die nichts als Böses im Sinne haben. Er weiß, daß seine einzige Chance, lebend aus dieser Welt zu fliehen, darin besteht, den Fluch umzudrehen. Nur durch das Sammeln von bestimmten Kristallen, die in sein Amulett passen, ist dies möglich.
Hiermit ist das Ziel von Troll eigentlich schon genannt: Wandern Sie durch zahlreiche Räume und sammeln Sie die Kristalle auf. Natürlich versuchen die Kobolde und allerlei anderes Ungetier, Sie an diesem Vorhaben zu hindern. Um Ihre Mission erfolgreich zu bestehen, sind Sie mit sehr eigentümlichen Waffen ausgerüstet. Um immer bereit zu sein, tragen Sie ständig eine gewisse Zahl von Löchern(!) in Ihren Taschen mit, um sie bei Bedarf einem Kobold vor die Füße zu werfen, der auf Nimmerwiedersehen darin verschwindet. Ein Kobold benötigt seinerseits ein Loch, um in den nächsten Raum zu kommen. Diese Art von Löchern trägt er aber nicht bei sich, sie müssen erst vom Oberzwerg Fjalar beschworen werden.
Die durch die Story des Spiels erweckten Hoffnungen werden jedoch durch seine Grafik enttäuscht. Auf dem Bildschirm präsentiert sich ein annähernd dreidimensionales Spielfeld, aufgebaut aus vielen einzelnen Plateaus. Die Figuren, also die Kobolde und der Troll, erscheinnen darauf fast unsichtbar klein. Hinzu kommt die zwar recht gelungene Farbgebung, die aber bei einer dermaßen klein geratenen Grafik das ganze noch unübersichtlicher macht. Die Animation erfolgt zwar recht schnell, ist aber trotzdem sehr ruckelig.
Da das Spielfeld sehr klein geraten ist, erweckt es den Anschein, als sei eine ganze Menge darauf los. Trotzdem fällt bei genauerem Betrachten auf, daß sich nicht mehr als acht Sprites auf dem Bildschirm befinden.
Lobenswert ist die Musikuntermalung. Während des Spielablaufs tönt ständig eine fetzige und hervorragend komponierte Melodie aus dem Lautsprecher. Dafür ist jedoch der Sound zu kurz gekommen, was allerdings kaum auffällt.
Die Steuerung macht das Game enorm schwierig. Leider wird sie unkorrekt und viel zu nervös abgefragt und umgesetzt. Dadurch ist es fast unmöglich, den Troll genau zu steuern. „Troll" ist dadurch fast unspielbar.
Das Interesse am Spiel verliert sich dadurch bereits nach kurzer Zeit. ● *mn*

Titel:	**Troll**	
Getestet:	**C-64**	
Umsetzungen:	**Spectrum**	
Im Test: ☒ Diskette ☐ Kassette	**Preis (DM):** 49,95	☒ **Joystick** ☐ **Tastatur** ☐ **Maus**

Wertung	**0**	**25**	**50**	**75**	**100**
Grafik					
Sound					
Bedienung					
Motivation					

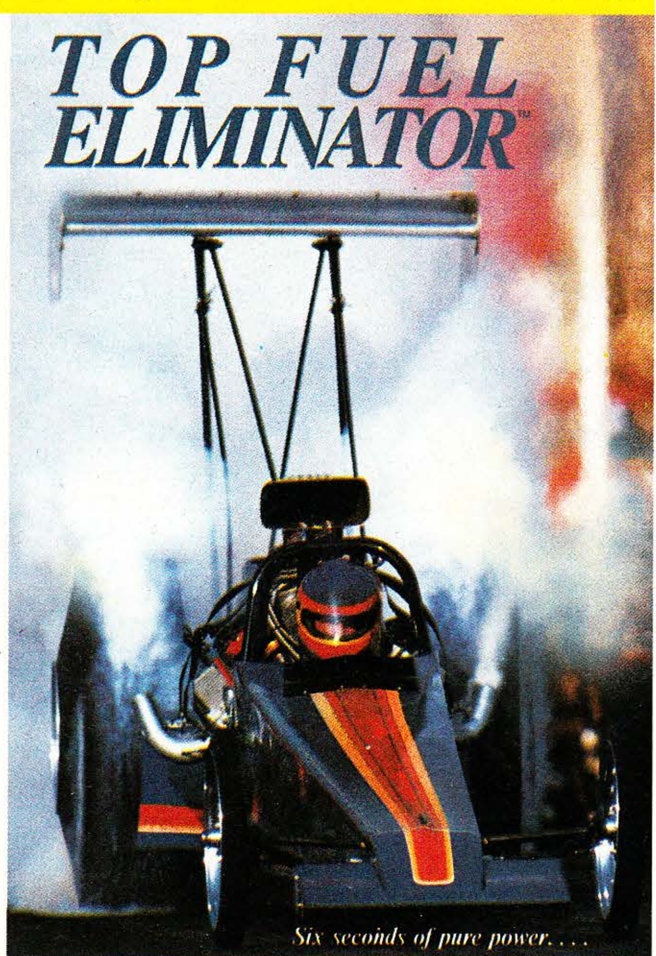
Top Fuel Eliminator

Ein neues Autorennspiel reißt inzwischen niemanden mehr vom Hocker. So war ich auch relativ abgeneigt, ein solches Spiel zu testen. Was sollte da schon "top" sein? Meine Meinung mußte ich aber sehr schnell ändern . . .

Sie befinden sich in einem Fahrzeug mit über 2500 PS unter der Haube. Der Wagen wurde speziell für Sie gebaut, um Sie in kürzestmöglicher Zeit zu einem 400 Meter entfernten Ziel zu bringen.

2500 PS unter der Haube

Ihre Reflexe müssen blitzschnell sein, denn dies ist wie das Steuern einer Rakete, die auf dem Boden entlanggleitet. Top Fuel Eliminator ist die Simulation einer ganzen Dragster-Rennsaison. Die Saison besteht aus neun Rennereignissen und wird auf der Basis eines einheitlichen Punktesystems bewertet.
Sie müssen gegen acht der besten Rennfahrer der Welt antreten. Um ein Rennen zu gewinnen oder sogar nur, um ohne größeren Schaden ans Ziel zu kommen, müssen Sie auch auf die Wetterbedingungen und den Fahrbahnzustand achten und Ihren Dragster entsprechend darauf einstellen. Fast wie im richtigen Leben.
Top Fuel Eliminator bietet Ihnen eine Vielzahl von Optionen an, mit denen Sie Ihre Gewinnchancen beeinflussen und erhöhen können. Unter anderem können Sie Ihren Flitzer mit verschiedenen Reifen oder sonstigen Zusatzgeräten ausrüsten. Tanken nicht vergessen, sonst geht sowieso nichts. Sind alle Vorbereitungen



getroffen, kann's losgehen. Das Steuern des Rennwagens ist wesentlich schwieriger, als ich zunächst angenommen hatte. Gerade als ich meinem Flitzer endlich unter Kontrolle gebracht habe, drängt mich ein Seitenwind an die Leitplanke. Also das Ganze nochmal von vorne. Diesmal klappts, mein Wagen passiert elegant

Übermut tut selten gut

die Ziellinie und kracht gegen die Zuschauertribüne. Aber macht nichts , passiert ist passiert. Auf zum Halb-Finale.
Im Laufe des Tages gelangte ich noch bis zur zweiten Rennstrecke, dann war mein Auto so kaputt, daß mich die Reparatur zu teuer gekommen wäre. Na, wenigstens durfte ich mich in der Highscore-Liste verewigen.

Fazit

Anfängern ist Top Fuel Eliminator nicht zu empfehlen, aber auch reine Action-Rennfreaks werden keine Freude an diesem Spiel haben. Es bietet so viele Optionen und Möglichkeiten, daß es ziemlich lange dauern kann, bis Sie sich damit zurechtfinden. Wenn Sie sich aber erstmal auskennen, werden Sie garantiert auf lange Zeit viel Spaß am Spiel haben. Und selbst wenn der Top Fuel Eliminator einmal in der Schublade landen sollte: An einem langen Winterabend wird er bestimmt wieder hervorgeholt werden, da er sich wohltuend von der Masse der Autorennspiele abhebt. Den Kauf bereut mit Sicherheit niemand. ●

TB

Titel:	Top Fuel Eliminator
Getestet:	C-64
Umsetzungen:	Apple II

Im Test:		Preis (DM):		
☒	Diskette	k.A.	☒	Joystick
☐	Kassette		☒	Tastatur
			☐	Maus

Wertung	0	25	50	75	100
Grafik					
Sound					
Bedienung					
Motivation					

Arkanoid II

Vor einem knappen Jahr erlebte die Breakout-Welle einen neuen Aufschwung. Der Grund hieß Arkanoid. Nun ist die Fortsetzung da: Arkanoid II.

Die Breakout-Welle reißt nicht ab. Es dürfte kein Computersystem mehr geben, für das nicht auch ein Breakout zu haben ist. Verständlich, denn schließlich macht diese Spielegattung sehr viel Spaß. Erstaunlich ist nur, daß sich Breakout-Versionen immer noch auf dem Softwaremarkt, der ja schon seit Jahren von außergewöhnlichen Spielideen verwöhnt wird, behaupten können. Besonders Imagine Software hat sich durch Arkanoid vor einem knappen Jahr einen guten Namen gemacht. Angenehm überrascht waren wir daher, als vor einigen Tagen der Nachfolger von Arkanoid in der Redaktion eintraf: Arkanoid II – Revenge of Doh. Das Spiel steht seinem Vorgänger in nichts nach. Lassen Sie sich deshalb warnen: Es macht süchtig.

Spannung und Faszination machen dieses Spiel zu einem heißbegehrten Game. Die Grafik unterstützt diese Aussage voll und ganz.

Breakout – ein Klassiker

Für Computerneulinge sei hier kurz das Breakout-Prinzip erläutert: Es gilt, mit einem Ball eine Mauer aus Einzelsteinen zu zerstören. Trifft der Ball auf einen Stein, so verschwindet dieser und der Spieler erhält einen Punkt gutgeschrieben. Der Ball trifft abwechselnd auf die Mauer und einen Schläger, der sich am unteren Bildschirmrand nach links und rechts bewegen lassen kann. Der Spieler übernimmt die Steuerung des Schlägers. Verfehlt er den Ball, wird ein Leben abgezogen.
Dasselbe Prinzip wird auch bei Arkanoid II angewandt.

Einmalige Features

Wie schon der Vorgänger weist diese Breakout-Variante einige Features auf, die das Spiel äußerst positiv von der Masse der Breakout-Versionen abheben. Einige der Mauerstücke lassen nämlich eine Art Pille zu Boden sinken, wenn sie vom Ball getroffen wurden. Nimmt der Spieler sie mit seinem Schläger auf, kann dies, je nach Farbe der Pille, unterschiedliche Wirkungen haben. Der Schläger kann verbreitert oder verschmälert werden, was die Treffsicherheit steigert beziehungsweise vermindert. Manchmal läßt sich der Schläger in eine Kanone umwandeln, mit welcher Sie die Steine einfach abschießen können. Das geht natürlich viel schneller. Ein besonderer Leckerbissen ist die Pille mit der Aufschrift „B". Durch sie können Sie nämlich problemlos ins nächste Level wechseln und gleichzeitig auch noch eine Menge Bonuspunkte einstreichen.

Viel Spaß durch viele Levels

Arkanoid II besitzt so viele Level, daß es ziemlich lange dauern kann, bis Sie alle gesehen haben. Damit sei wenigstens eine lang anhaltende Spielmotivation garantiert, werden Sie jetzt sicher denken. Viel besser – dieses Spiel macht süchtig. So unwahrscheinlich es auch klingen mag, aber wer einmal Arkanoid II gespielt hat, kommt so schnell nicht mehr vom Computer los. Wir in der Redaktion saßen stundenlang gebannt vor den Monitoren. Wenn uns vor einem Monat jemand gesagt hätte, daß uns ein Breakout so faszinieren würde, hätten wir ihn wohl mit einem Lächeln abgespeist. Nun, Arkanoid II – Revenge of Doh hat uns eines Besseren belehrt.

Fazit

Das Game kann jedem, ob Ballerfreak oder Arcade-Champ, uneingeschränkt empfohlen werden. Am besten, Sie reservieren sich gleich noch heute ein Exemplar bei Ihrem Computer-Fachhändler, sonst kann es leicht passieren, daß Ihnen ein anderer Arkanoid-Süchtiger zuvorkommt. ● *TB*

Titel: Arkanoid II
Getestet: C-64
Umsetzungen: Schneider CPC, Amiga

Im Test: ☒ Diskette ☐ Kassette
Preis (DM): 39.–
☒ Joystick
☒ Tastatur
☐ Maus

Wertung	0	25	50	75	100
Grafik					
Sound					
Bedienung					
Motivation					

Rimrunner

Immer neue Katastrophen bahnen sich in den hintersten Ecken des Universums an. Immer neue machthungrige Herrscher greifen in das friedliche Alltagsleben idyllischer Planeten und Monde ein. So auch bei Rimrunner von Palace Software.



Folge 4.357 unserer Science-Fiction-Serie handelt von den armen Ameisigen, die ständig von den bösen Rickrakkern angegriffen werden. Um ihre Kolonien zu schützen, verwenden die Ameisigen Kraftschilder, mit denen sie große metallene Gegenstände abwehren können, so auch die Rickracker-Sturmtruppen.

Die Kraftgeneratoren, die die Schutzschilde für lange Zeit aufrechterhalten, brauchen ungeheure Energiemengen, die ihnen ständig zugeführt werden müssen. Dafür sind die Randpatrouillen verantwortlich, die am äußersten Rand einer Kolonie eingesetzt werden und mit ihren schnellen Reit-Reptilien die Generatoren aufladen und allerlei Kleingetier, das die Schilde durchbrochen hat, vernichten.

Fleißig Ballern ist angesagt

Dieser Patrouille gehören auch Sie an. Sie haben vier Minisaurier zur Verfügung, die Sie nach Bedarf verwenden können. Natürlich darf auch eine Schußwaffe nicht fehlen, mit der Sie unter den Aliens aufräumen. Wie sich herausgestellt hat, gibt es eine ganze Menge Eindringlinge.

Nach einem kurzen Ladevorgang erscheint ein mittelprächtiges Titelbild zum Spiel, das einen Ameisigen auf seinem Reitsaurier zeigt. Wollten wir das Game nach seinem Bild beurteilen, so fiele die Wertung nicht sehr gut aus. Glücklicherweise ist das nicht nötig und wir wollen weiterladen. Leider ergeben sich bei uns ab hier ständige Ladefehler, die zum Absturz des Systems führen. Diese Fehler müssen aber nicht bei jeder Version auftreten. Trotzdem sollten Sie sich vor dem Kauf vergewissern, daß der Händler ein schadhaftes Spiel zurücknimmt.

Ist das Game komplett in den Speicher geladen, so erscheint eine gut aufgemachte Highscore-Liste. Erst nach Tastendruck lädt der hervorragend animierte Ameisige sein Gewehr durch und das Spiel kann beginnen.

Der Spielbildschirm bei Rimrunner besteht aus einem bewegten Vordergrund, auf dem die Hauptfigur mit seinem Reittier nach links oder rechts reiten kann. Im Hintergrund scrollt die Silhouette einer Weltraumkolonie fleißig den Bewegungen des Sauriers nach.

Während Sie Ihre Figur über den Planeten lenken, fällt allerlei Zeug vom Himmel, das Sie abschießen sollten. Ab und zu schwebt Ihnen ein Generator entgegen. Laden Sie ihn auf, denn das ist das Ziel des Spiels. Da das nicht vom Reittier aus geht, haben Sie die Möglichkeit, abzusitzen und zu Fuß zu gehen.

Sowohl die Grafik des Sauriers als auch die des Reiters ist gut gelungen, sauber animiert und sehr farbenfroh. Die Aliens sind zwar sauber konstruiert, allerdings nur in den seltensten Fällen nennenswert animiert.

Der Vorder- und Hintergrund scrollt fließend in horizontale Richtungen. Da sich der Hintergrund langsamer bewegt, entsteht ein gewisser Tiefeneffekt. Auch die Gebäude sind sauber gezeichnet und glänzen durch ihre Farbenfreude.

Außer den gut gelungenen Grafikeffekten hat das Spiel noch mit einigen hervorragenden Musikstücken aufzuwarten. Besonders der

Soundtrack der Highscoreliste läßt keine Wünsche mehr offen. Aber auch die Hintergrundgeräusche wurden gut programmiert. Sowohl die Explosionen und Schußgeräusche als auch der Pfiff des Ameisigen, der sein Reittier zu sich ruft, tönt ansprechend aus dem Lautsprecher.

Altmodisches Spielprinzip

Trotz dieser vielen Pluspunkte fehlt es Rimrunner an etwas sehr Wichtigem: der Originalität. Das Spielprinzip ist absolut nichts Neues, ganz im Gegenteil: Die Programmierer scheinen in die Anfangszeit des Commodore 64 zurückgekehrt zu sein. Allerdings nur in ihren Ideen, nicht in den Programmiertechniken.
Durch die altmodische Spielidee flaut die anfängliche Begeisterung über Grafik und Sound schnell wieder ab und verwandelt sich in Langeweile.

Fazit

Hätte Rimrunner nicht diese antiquierte Spielidee, wäre ein erstklassiges Actiongame daraus geworden. In Sachen Grafik und Sound hat das neue Palace-Produkt nämlich eine ganze Menge zu bieten. So aber wird es schnell langweilig, da sich die gesamte Handlung auf hirnloses Ballern beschränkt. Fans solcher scrollenden Ballergames kommen jedoch auf ihre Kosten. ● *mn*

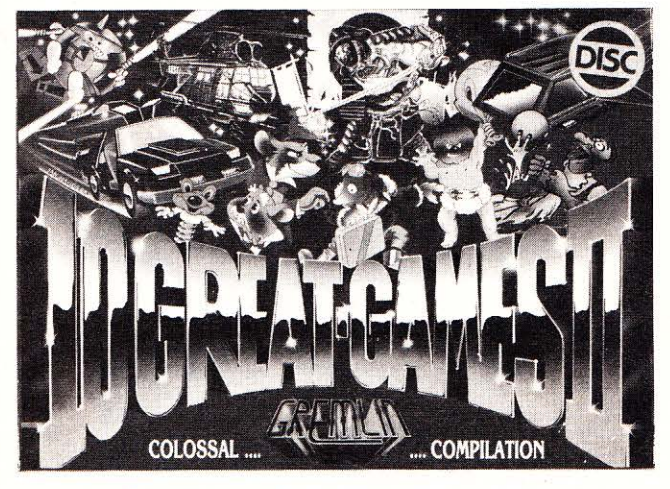
Titel: Rimrunner
Getestet: C-64
Umsetzungen: Schneider CPC, Atari ST

Im Test: [X] Diskette [ ] Kassette
Preis (DM): 49,90
[X] Joystick
[ ] Tastatur
[ ] Maus

Wertung 0 — 25 — 50 — 75 — 100
Grafik
Sound
Bedienung
Motivation

10 Great Games II

Vor einem guten halben Jahr überraschte der Engländer Gremlin mit einem Software-Sampler für den C64. „10 Great Games" eroberte relativ schnell die Hitparaden. Jetzt ist der Nachfolger da.



Software-Sampler, auch Compilations genannt, sind eine tolle Sache. Der Käufer erwirbt in der Regel fünf bis zehn Spiele zum Preis von einem. Die Spiele sind deshalb aber keineswegs schlecht oder seit langem aus der Mode, sondern lediglich nicht mehr die neuesten auf dem aktuellen Software-Markt. Oft lohnt sich der Kauf eines solchen Samplers wesentlich mehr als der eines schlechten aktuellen Spieles. So auch bei „10 Great Games II" von Gremlin.
Auf drei Kassetten beziehungsweise Disketten befinden sich, wie der Name schon sagt, zehn Spiele für den C64. Ob es auch Umsetzungen für andere Computersysteme geben wird, ist noch nicht bekannt.
Auf Kassette beziehungsweise Diskette eins finden Sie die Spiele Water Polo, Rebounder, Convoy Raider und Samurai Triology. Besonders hervorzuheben ist Water Polo. Sie sind eingeladen, an einer Partie Wasserball teilzunehmen. Verhelfen Sie Ihrer Mannschaft zu Sieg und Ruhm. Gute Grafik und flotter Sound unterstützen Sie dabei.
Auf Datenträger zwei gibt's die Fortsetzung des Klassikers Jack The Nipper. Diesmal muß Jack in komplizierten Höhlensystemen seine Bosheit unter Beweis stellen.
Wer den gleichnamigen Kinofilm gesehen hat, darf sich Basil, The Great Mouse Detective nicht entgehen lassen. Lösen Sie einen schwierigen Fall in Londons schäbigen Hafenstraßen und toten Wasserkanälen.
Wenn Sie sich statt mit ungelösten Rätseln lieber mit brutalen Gegnern herumschlagen, dann ist Mask genau richtig für Sie. Bekämpfen Sie mit ihren Truppen die bösen und trügerischen Venoms.
Auf Datenträger Nummer drei finden sich Bulldog, Thing Bounces Back und Auf Wiedersehen Monty. Da die drei Programme noch relativ aktuell sind, verzichte ich auf eine nähere Beschreibung.

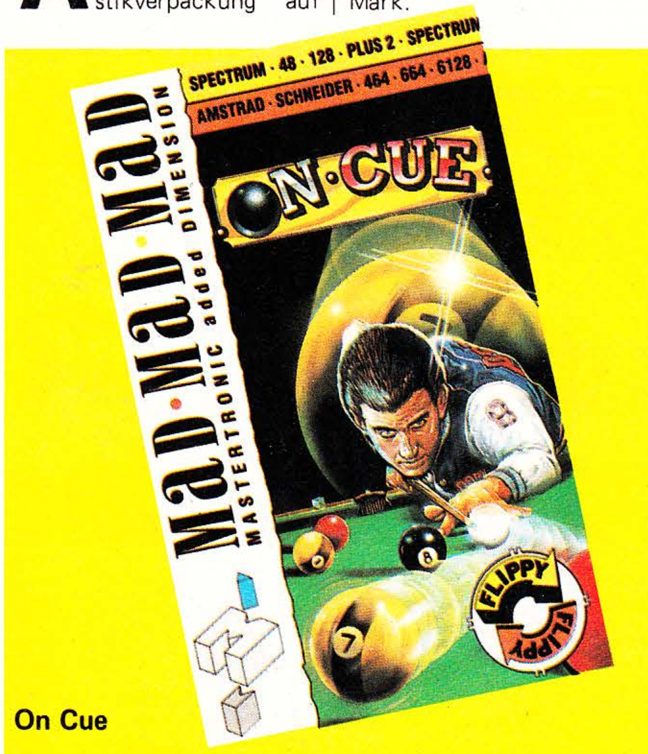
Fazit

Wer die einzelnen Spiele noch nicht besitzt oder kennt, sollte so schnell wie möglich zugreifen. Und wer ein paar Mark übrig hat, kann sich den Vorgänger „10 Great Games" auch noch anschauen. ● *TB*

Games für kleine Geldbeutel

Daß Spiele nicht unbedingt teuer sein müssen, bewies uns wieder einmal die Firma Mastertronic. Lesen Sie selbst, womit der bekannte Software-Hersteller diesen Monat aufwartet.

Alle Spiele werden in der gewohnten Plastikverpackung auf Kassette geliefert. Sie kosten zwischen 10 und 15 Mark.

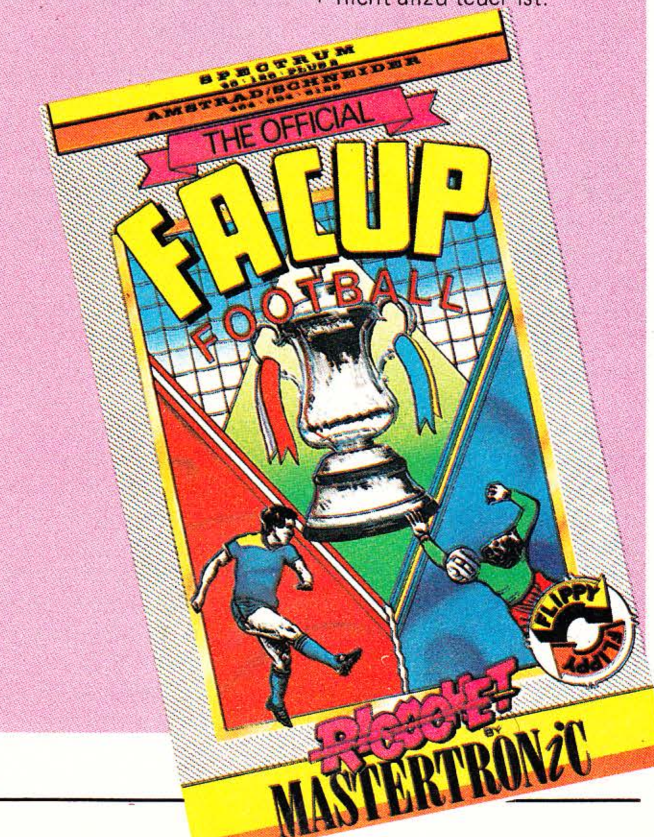


On Cue

Wer eine preiswerte Billiard-Simulation sucht, sollte sich bei seinem Fachhändler nach On Cue erkundigen. Gute Grafiken und eine flotte Hintergrundmusik lassen einer flotten Billiard-Partie nichts mehr im Wege stehen. Es können zwei Spieler gegeneinander antreten, nicht gerade eine Selbstverständlichkeit bei Simulationen dieser Art. Natürlich können Sie auch allein gegen den Computer spielen. Allerdings ist dieser ein so ausgezeichneter Spielpartner, daß Sie spätestens nach zwei verlorenen Partien die Lust verlieren werden.
Gesteuert wird On Cue mit dem Joystick oder über die Tastatur. Die Schneider-CPC-Besitzer sollten einen Farbmonitor ihr eigen nennen. Zu haben ist das Game für den Schneider CPC und den Spectrum.

The Official FA Cup Football

Ebenfalls für CPC und Spectrum wird das offizielle Spiel zum Fußball-World-Cup angeboten. Sie müssen eine oder mehrere Mannschaften managen, das heißt Begegnungen arrangieren, Spieler ein- und verkaufen und vieles mehr. Vom Aufbau her erinnert dieses Game an den legendären Football Manager. Auch FA Cup besitzt keine richtige Grafik, an Sound ist lediglich ein Piepston zu hören.
Bis zu acht Spieler können teilnehmen. Allerdings wirkt sich eine hohe Spielerzahl negativ auf die Motivation aus, da man länger warten muß, bis man wieder an die Reihe kommt. Im großen und ganzen kann ich Ihnen von diesem Game nur abraten, denn der Spielablauf ist recht eintönig. Wer jedoch an Football Manager Gefallen gefunden hat, kann sich das Game ja mal ansehen, zumal es mit 15 Mark auch nicht allzu teuer ist.

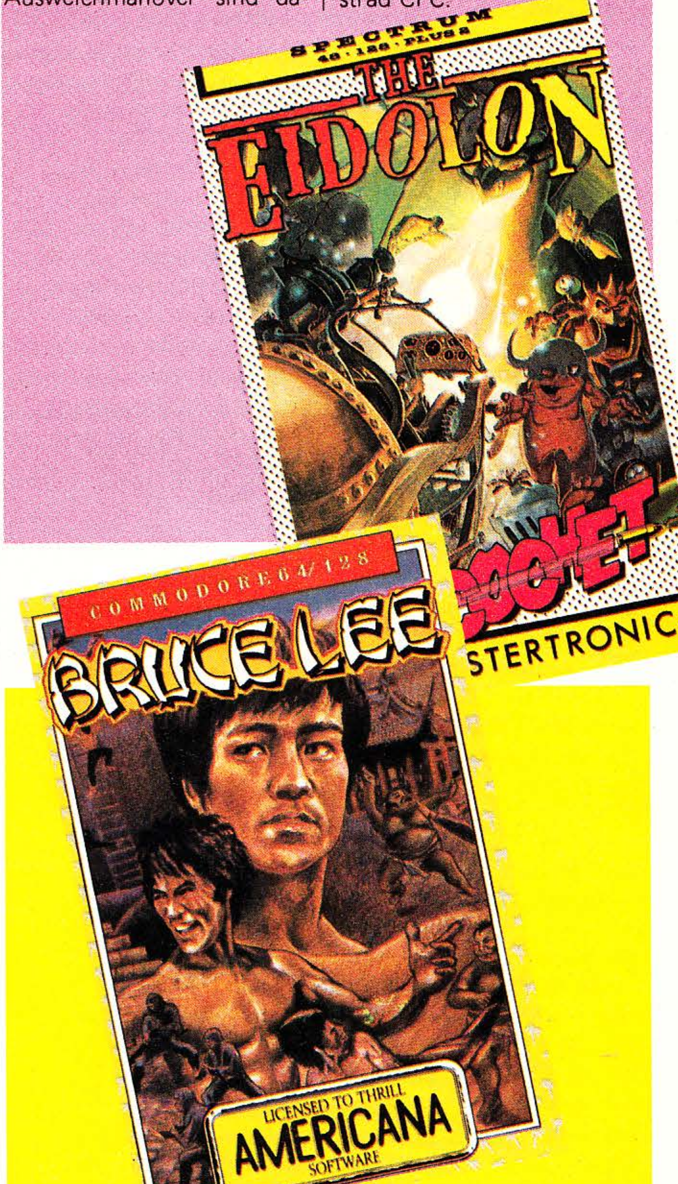


Impossible Mission

Mit diesem Game hat Mastertronic, wie in letzter Zeit schon oft, die Vertriebsrechte für einen Oldie erworben, den man allerdings besser als Goldie bezeichnen sollte. Erinnern Sie sich zwei Jahre zurück. Ein verrückter Professor ist aus dem Irrenhaus ausgebrochen und will mal wieder die Welt erobern. Zu diesem Zweck baut er sich ein unterirdisches Laboratorium, das er von Robotern bewachen läßt. Im Inneren seiner Höhle schmiedet er einen grausamen Plan zur Vernichtung der Menschheit. Nur ein Mann kann ihn stoppen: Elvin Atombender, dessen Rolle Sie übernehmen dürfen. Dringen Sie in das unterirdische Reich des Wahnsinnigen ein, zerstören Sie seine Apparaturen und nehmen Sie ihn gnadenlos fest. Genauso spannend wie die Story klingt, ist auch das Spiel. Von der Grafik her gehört Impossible Mission trotz seines Alters immer noch zur Spitzenklasse. Alle Sprites sind sehr schön animiert und lassen kaum Wünsche offen. Zur aktuellen Spielsituation erklingen diverse Hintergrund-Geräusche, auf manchen Systemen sogar digitalisiert. Impossible Mission können Sie künftig mit dem Joystick auf dem C64 und dem Schneider CPC spielen.

The Eidolon

Vor gut 18 Monaten war The Eidolon noch in den Software-Top-Ten zu finden. Auch für dieses Spiel hat Mastertronic nun die Rechte aufgekauft und vertreibt es künftig als Billigspiel. Ihre Aufgabe ist es, mit einem Raumgleiter durch ein unterirdisches Höhlensystem zu fliegen und dabei magische Juwelen aufzusammeln. Diese benötigen Sie, um von einem Level ins nächste zu gelangen, denn am Ende jedes Levels wartet ein feuerspeiender Drache, der magische Juwelen überhaupt nicht verträgt. Auf Ihrem Höhlenflug werden Ihnen viele Gesellen begegnen, die Ihr Raumschiff zerstören wollen. Gewagte Ausweichmanöver sind daher an der Tagesordnung. Gutgemachte dreidimensionale Grafiken und flotte Soundeffekte lassen The Eidolon zu einem Vergnügen werden. Das Spiel gibt's für C64 und Schneider/Amstrad CPC.



Bruce Lee

Über dieses Spiel brauche ich wohl keine großen Worte zu verlieren. Ihre Aufgabe ist es, als Bruce Lee in den magischen Tempel einzudringen und diverse Gegenstände aufzusammeln, die Punkte einbringen. Bruce Lee ist der Urahn der Jump-and-Run-Spiele, macht aber immer noch viel Spaß. Seinen Preis von 15 Mark ist es allemal wert. Bruce Lee ist für C64, Schneider/Amstrad und Spectrum erhältlich.

Fazit

Auf dem Billigsoftware-markt tut sich einiges, wie man sieht. Die LOAD & RUN-Redaktion bleibt am Ball. ● *TB*

Kurzberichte

In diesem Monat git es wieder für jeden Rechner Neuigkeiten auf dem Spielesektor. Da im letzten Spiele-Magazin ja nur die Games für den Commodore 64 betrachtet wurden (weil keine anderen verfügbar waren), sind in diesem LOAD & RUN auch wieder Spiele für Atari ST und Amiga vertreten.

Auf dem Atari ST gibt es diesen Monat – außer den im Spieleteil ausführlich getesteten Games – nur zwei nennenswerte Neuheiten: Trauma und Bob Moran IV, beide von der französischen Softwarefirma Infogrames.

Trauma: Im Weltraum nichts Neues

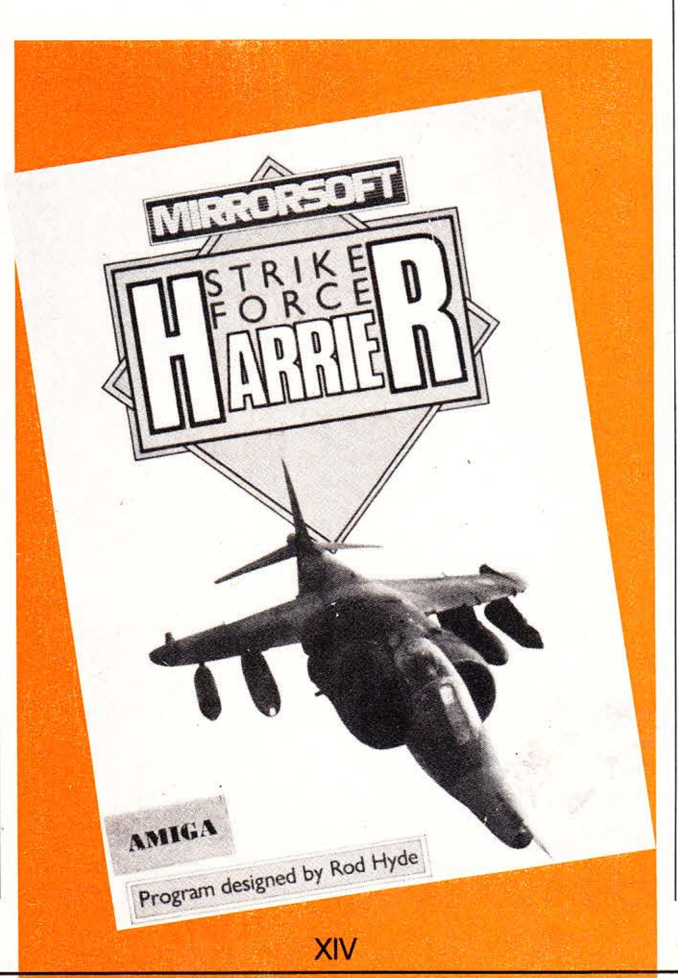
Bei Trauma handelt es sich wieder einmal um ein wüstes Weltraumballerspiel, wie es sie schon zu hunderten gibt. Nach Angaben des Herstelers soll jedoch etwas ganz Besonderes an ihm dran sein.

Bob Moran: Abenteuer unter Wasser

Bob Moran IV ist bereits der vierte Teil der Helden-Serie. Nach den drei Vorgängern mit den Titeln Rittertum, Im Jungel und Science Fiction spielt der neueste Teil tief unter der Meeresoberfläche. Auch hier hat der Held wieder gegen übermächtige Feinde zu kämpfen und ist wie immer mit supermodernen Geräten und Waffen ausgerüstet.
Die Atari-Version ist ganz akzeptabel, bei der Commodore 64-Adaption bleibt abzuwarten, ob sie genauso ist, wie bei Teil I.

Für den Amiga gibt es zwei Neuheiten, die eigentlich nicht so ganz neu sind.

Strike Force Harrier wurde vom Schneider- und Commodore-Rechner auch auf den Amiga adaptiert. Heraus kam eine mittelmäßige Mischung aus Flugsimulator und Aktiongame, die einem richtigen Simulator wie Jet nicht einmal annähernd das Wasser reichen kann. In Sachen Geschwindigkeit bleibt zwar kein Wunsch offen, technisch gesehen unterscheidet den Simulator von einem primitiven Ballerspiel mit einigen Extras nicht mehr viel. Alle nötigen Funktionen sind vorhanden, aber auch nicht mehr. Außerdem fehlt bei Simulator die Vorstellung des Fluggefühls. Die Landschaft fiel sehr monoton aus. Außer einigen Feldern und wenigen Bergen (Pyramiden) hat das Programm nichts zu bieten. Auf dem Commodore 64 wäre Strike Force Harrier eine Wucht, aber auf dem Amiga müssen schon andere Maßstäbe gesetzt werden.



Witzball: Wie auf dem Commodore64

Eine andere Adaption nennt sich Witzball. Auf dem 64er schon als Superspiel bekannt, wurde es nun auf den Sechzehn-Bitter übernommen – und enttäuscht ebenfalls ein wenig. Es sind kaum Veränderungen gegenüber der Commodore-Version feststellbar. Der Sound ist kläglich. Außer einer Titelmelodie tönt nur ein dumpfes Aufprall- und Ballergeräusch aus dem Lautsprecher. Die Grafik wurde nur wenig verbessert, ist aber trotzdem akzeptabel. Vom Spielwitz ging nichts verloren, da auch das C64-Game eine Menge davon besaß. Trotz der wenigen Verbesserungen ist Witzball also immer noch sehenswert – auch auf dem Amiga.

Iznogud: Eine Katastrophe für sich

Ebenfalls drei neue Games gibt es für den Commodore64.
Iznogud ist ein weiteres Spiel von Infogrames, das wir wegen Qualitätsmangels nicht ausführlich testen wollten. Es lehnt sich an die bekannte Comicserie vom Kalifen an und ist wieder einmal ein reines Geschicklichkeitsspiel. Qualitativ ist es kaum eine Erwähnung wert. Ein derart unerkennbares Durcheinander von verschiedenen Pixeln, die Sprites darstellen sollen, bekommt man selten geboten. Selbst das vor einigen Ausgaben angesprochene Ninja Hamster ist nichts dagegen. Auch der Sound ist eigentlich keine Erwähnung wert. Iznogud ist ein völlig danebengeratenes und unspielbares Game. Qualitativ eine Katastrophe.

Ein Strategiespeil namens Patton versus Rommel veröffentlichte Electronic Arts.

Patton Versus Rommel: Wüste(r)n Krieg

Das Game ist spieltechnisch sehr komplex aufgebaut. Darunter hat die Programmierung allerdings nicht wenig gelitten. Die Grafik ist teilweise nur sehr schwer zu erkennen und dazu langsam bewegt, obwohl nur Icons über eine Landkarte bewegt werden. Überhaupt läßt die Programmgeschwindigkeit etwas zu wünschen übrig. Von Electornic Arts ist der Bildschirm-Spider eigentlich Besseres gewöhnt. Da das Strategiespiel zudem den Afrikakrieg im Zweiten Weltkrieg simuliert, ist Patton versus Rommel eher ein Fall für die Bundesprüfstelle als für einen Testbericht.

Starfleet: VC-20-Zeiten im Weltraum

Aber Electronic Arts schlug nocheinmal mit einem Simulations- und Strategiespiel zu. Und wieder ging der Schlag daneben. Starfleet ist für den Commodore64 und Amiga erhältlich. Wir besprechen jedoch nur den CBM-Version, da sie der Amiga-Variante fast gleicht — oder umgekehrt. Jedenfalls erweckt Starfleet den Eindruck, als wäre für den Amiga zu diesem Game gleich auch ein C64-Emulator mitgeliefert worden — der Einfachheit halber. So ähnlich sind die beiden Versionen.
Das Game macht den Eindruck, als sei es in purem BASIC geschrieben, so primitiv und langweilig ist es. Es läßt sich am ehesten mit einem etwas komplizierten „Schiffeversenken im Weltraum" beschreiben. Der Bildschirm ist nur in ein Textfenster, ein Rasterfeld und ein Statuswindow unterteilt. Das Feld ist sehr klein geraten und die Symbole darauf sind derart winzig, daß sie schon gar nicht mehr ins Auge fallen. Die Schrift ist in normalem Zeichensatz geschrieben und wird einer Art und Weise gedruckt, wie wir sie aus den BASIC-Zeiten des Commodore kennen. Der Spielwitz hält sich das ganze Spiel über auf einem Niveau um Null. Das Game nützt den Rechner nie und nimmer aus und ist auf dem Amiga und dem Commodore gleich schlecht. Bleibt zu hoffen, daß die beiden Strategiespiele ein einmaliger Ausrutscher bei der sonst so guten Produktreihe von Electronic Arts bleiben. Ein schlechter Monat für das Edel-Softwarehaus. ● *mn*

Computer Classics

Neben dem ebenfalls in dieser Ausgabe vorgestellten Spiele-Sampler 10 Great Games traf von Ariolasoft eine weitere Compilation ein: Computer Classics für den Commodore C64 (128 im 64er Modus).

Auf der Diskette finden Sie fünf Spiele, die zwar nicht mehr ganz neu, aber trotzdem noch ihr Geld wert sind. Vor allem für Ein- und Umsteiger bieten solche Software-Sampler ja bekanntlich viel Spiel für wenig Geld.

Dynamite Dan

Das seinerzeit äußerst beliebte Jump-and-Run-Spiel Dynamite Dan verführt Sie in die Welt des bösen Doktor Blitzen, der einen Plan zur Eroberung der Erde entworfen hat. Sie als der Held Dynamite Dan müssen diesen Plan stehlen. Da das Spiel über 30 verschiedene Screens hat, kann die Suche unter Umständen sehr viel Zeit beanspruchen. Dadurch wird die Sache nicht so schnell langweilig.

Into The Eagles Nest

Im Action-Adventure Into The Eagles Nest müssen Sie Gefangene aus dem Lager des Feindes befreien. Aber Vorsicht, denn der Gegner schläft nicht und hat seine gesamte Armee mobilisiert, um Ihren Helden zu vernichten. Es darf geballert werden. Grafik und Sound sind nicht gerade überragend, aber dennoch ausreichend.

Zynaps

Ein weiteres Ballerspiel ist Zynaps. Hier dürfen Sie sich so richtig austoben. Mit den modernsten Waffen ausgerüstet, müssen Sie das Lager der feindlichen Aliens zerstören. Gute Grafiken und eine flotte Hintergrundmusik lassen so schnell keine Langeweile aufkommen.

The Pumpkin Strikes Back

Die böse Hexe hat all Ihre Kürbiskumpane vernichtet. Sie, der letzte überlebende Kürbis, haben Rache geschworen. Dringen Sie ins verwünschte Schloß der Hexe ein, finden Sie die Übeltäterin und schneiden Sie ihr eine Haarsträhne ab, aus der anschließend ein Zaubertrank gebraut werden muß. Die Grafik von Cauldron II ist sehr liebevoll gestaltet und auch der Sound läßt kaum Wünsche offen. Meiner Meinung nach ist Cauldron II das gelungenste Spiel auf der Diskette.

Aliens

Die Aliens kommen. Bei diesem Spiel handelt es sich um das offizielle Computerspiel zum Kinofilm Aliens, der bei uns vor einigen Jahren die Kassen füllte. Sie als Officer Ripley müssen die Außerirdischen aufhalten — nicht gerade ein Zuckerlekken. Spaß macht Aliens auf jeden Fall, auch wenn die Grafik bei der C64-Version nicht sonderlich gut gelungen ist. *TB*

Player's Pages

Hallo, Spielefreaks und Joystickartisten! Willkommen bei den Players Pages des LOAD & RUN-Spielemagazins. Wer Tips, Tricks, Lösungen oder Karten zu jedem x-beliebigen Spiel für jedes x-beliebige System hat oder sucht, der ist hier genau richtig.

Zuerst mal ein dickes Lob an die MSX-User, die sich diesmal für die Players' Pages besonders ins Zeug gelegt haben.

MSX-Spielepokes

Von unserem Leser Dag Frommhold aus Neuffen stammen die folgenden kleinen Poke-Programme für den MSX-Computer. Einfach abtippen, starten und staunen.

```
 5 'Hunchback the Adventure
10 BLOAD"CAS:"
20 POKE –28370,0
30 DEFUSR=&H9000:A=USR(0)

10 ' Turmoil
20 LOAD "CAS:"
30 POKE &HEC1,&H0
40 DEFUSR=&HE646:A=USR(0)

10 ' Mutant Monty
20 CLEAR 100,&H87FF
30 BLOAD "CAS:"
40 POKE –27872,0
50 DEFUSR=&H9000:A=USR(0)
```

Dag hat auch beim Spiel Valkyr einen Cheat-Mode entdeckt, in den Sie durch gleichzeitiges Drücken von TAB, CTRL, SHIFT, ESC, CAPS, CURSOR UP, DOWN und LEFT, allerdings erst im Hauptmenü, gelangen. Als kleine Zugabe noch die Keywörter für The Goonies: GOONIES, MR SLOTH, GOON DOCKS.

Masters Of The Universe

Unser Leser Hanno Schehly hat uns für alle Masters of the Universe einen Stadtplan zugeschickt, auf dem die magischen Chords markiert sind. Jetzt dürfte es wirklich kein Problem mehr sein, das Spiel zu beenden.

Alles für den CPC

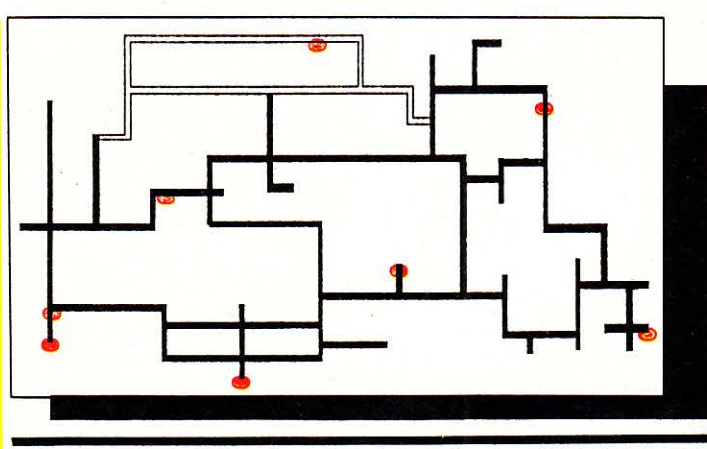
Markus Sledzinski hat sich für die CPC-User in der Spielewelt umgesehen – mit Erfolg, wie Sie an der nachfolgenden Poke-Liste sehen können.

SIGMA SEVEN	POKE 34432, anzahl
COPOUT	POKE &13A,&FF
NEMESIS	POKE &33C,&00
PROJECT FUTURE	POKE &9B9C,&08
ANDROID ONE	POKE &7391,&00
SPINDIZZY	POKE &A85E,&C9
SHOCKWAY RIDER	POKE &7D8E,&00
KILLAPEDE	POKE &4A14,&00
IMP. MISSION	POKE &12BF,&C9
WIZBALL	POKE &7CC2,&00
BLAGGER	POKE &7CC2,&00 POKE &7CC4,&00
DRUID	POKE 11265,00 POKE 8449,195
MIKIE	POKE 25116,195
BOMBSCARE	POKE &A1F0, &FF
ARMY MOVES (1)	POKE &0B3E,&00
ARMY MOVES (2)	POKE &0865,&00
CLASSIC AXIENS	POKE &478C,&7F

Wer weiß mehr?

Auch diesmal sind wieder viele Fragen zu beantworten. Wer kann weiterhelfen? Eine ganze Menge Probleme hat Hanno Schehly. Bei Jack The Nipper findet er die Diskette nicht. Wie kann man den Leim verwenden? Beim Spiel Nightmare findet Hanno die Exit-Steine nicht und für Molecule Man benötigt er einen Plan. Auch für das Spiel Die Erbschaft wird dringend Hilfe benötigt. Wer nimmt im ersten Teil welchen Gegenstand und was muß man im Flughafen (Teil 2) machen? Wer Hanno helfen kann, möge uns bitte schreiben.

Euer
Thomas Bosch



Von ihrem Ausgangspunkt betrachtet, müssen Sie versuchen möglichst ohne Umwege, ihr Ziel zu erreichen. Hier ist der Lageplan des Labyrintes aus dem Spiel Heman

PROGRAMMSERVICE

Hiermit bestelle ich in Kenntnis Ihrer Verkaufsbedingungen die Listings dieses Heftes auf

□ **Kassetten zu 40,-** □ **Disketten zu 40,- (16er)**

Ich zahle:
Zutreffendes bitte ankreuzen!
per beigefügtem Scheck ( ) Schein ( )

Gegen Bankabbuchung am Versandtag ( )
Meine Bank (mit Ortsname) ______ **16/III**
Meine Kontonummer ______
Meine Bankleitzahl ______ (steht auf jedem Bankauszug) ______
Vorname ______ Nachname ______
Str./Nr. ______ Plz/Ort ______

Verkaufsbedingungen: Lieferung nur gegen Vorkasse oder Bankabbuchung. Keine Nachnahme. Umtausch bei Nichtfunktionieren. **16/III**

Unterschrift ______

Bitte ausschneiden und einsenden an

COMMODORE WELT
KASSETTENSERVICE **16/III**
POSTFACH 1161
D-8044 UNTERSCHLEISSHEIM

LESER WERBEN LESER

GEWINNEN SIE EINE COMPUTER-UHR! Und zusätzlich eventuell noch ein großes Commodore-Buch. Oder ein Paket Disketten. ODER AUCH EINEN COMMODORE-DRUCKER – ODER EINE DISKETTENSTATION! Wie? Sie werben einen Abonnenten. Dann haben Sie auf jeden Fall schon die Computer-Uhr gewonnen. Zusätzlich verlosen wir unter allen, die mitmachen, jeden Monat vier weitere wertvolle Preise. Und alle sechs Monate gibt es einen Hauptpreis unter allen Abo-Werbern zu gewinnen. Also: Mitmachen. Mitgewinnen.

Ja, ich mache mit beim Abo-Wettbewerb. Ich habe



Herrn/Frau ______

______

Straße/Hausnr. ______

______

Plz/Ort ______

______

als neuen Abonnenten der C16/P4 geworben.

Der neue Abonnent war bisher noch nicht Bezieher dieser Zeitschrift.
Als Prämie erhalte ich nach Eingang des Abo-Entgeltes auf jeden Fall eine Computer-Uhr, wie abgebildet, und nehme zusätzlich noch an der Verlosung des Monats sowie der halbjährlichen Hauptpreise teil. Mir ist bekannt, daß der Rechtsweg bei den Verlosungen ausgeschlossen ist.
Meinen Preis senden Sie an

Name ______

Straße/Hsnr. ______

Plz/Ort ______

(Bitte ausschneiden und zusammen mit der Abo-Bestellkarte links einsenden!) **16/III**

VERDIENEN SIE GELD MIT IHREM COMPUTER!

Haben Sie einen Commodore VC 20 oder C 64? Einen 16/116, Plus 4? Oder einen 128? Können Sie programmieren? In Basic oder Maschinensprache? Dann bietet COMMODORE-WELT Ihnen die Möglichkeit, mit diesem Hobby Geld zu verdienen!

Wie? Ganz einfach. Sie senden uns die Programme, die Sie für einen Abdruck als geeignet halten, zusammen mit einer Kurzbeschreibung, aus der auch die verwendete Hardware – eventuelle Erweiterungen – benutzte Peripherie – hervorgehen muß (Schauen Sie sich dazu den Kopf unserer Programmlistings an.)

Benötigt werden: Zwei Listings des Programms sowie eine Datenkassette oder Diskette! Wenn die Redaktion sich überzeugt hat, daß dieses Programm läuft und sich zum Abdruck eignet, zahlen wir Ihnen pro Programm je nach Umfang bis zu DM 300,–!

Sollten Sie keinen Drucker haben, genügt der Datenträger.

Sie erhalten Ihre Kassette/Diskette selbstverständlich zurück, wenn Sie einen ausreichend frankierten Rückumschlag mit Ihrer Adresse beifügen.

Bei der Einsendung müssen Sie mit Ihrer Unterschrift garantieren, daß Sie der alleinige Inhaber der Urheber-Rechte sind! Benutzen Sie bitte anhängendes Formular! (Wir weisen darauf hin, daß auch die Redaktion amerikanische und englische Fachzeitschriften liest und „umgestaltete" Programme ziemlich schnell erkennt).

Um Ihnen die Arbeit zu erleichtern, finden Sie hier ein Formular. Sie können es ausschneiden oder fotokopieren.

Name des Einsenders: ______________________________
Straße/Hausnr./Tel.: ______________________________
Plz/Ort: ______________________________

Hiermit biete ich Ihnen zum Abdruck folgende(s) Programm(e) an:

____
____
____
____

Benötigte Geräte: ______________________________
____

Beigefügt ( ) Listings ( ) Kassette ( ) Diskette

Ich versichere, der alleinige Urheber des Programmes zu sein!
Hiermit ermächtige ich die Redaktion, dieses Programm abzudrucken und wirtschaftlich zu verwerten. Sollte es in den Kassetten-Service aufgenommen werden, erhalte ich auch dafür eine entsprechende Vergütung, das Copyright geht insoweit auf den Verlag über.

____
Rechtsverbindliche Unterschrift

COMMODORE WELT
PROGRAMM-REDAKTION
POSTFACH 1161
D-8044 UNTERSCHLEISSHEIM

Fortsetzung von Seite 62

Das Zero-Flag

Wissen Sie, warum viele Programmierer das Ende eines Datensatzes mit einer Null kennzeichnen?
Laden Sie den Akku oder irgend ein anderes Register mit einer von Null verschiedenen Zahl, dann mit der Zahl null. Beim Beobachten des Statusregisters bemerken wir eine Veränderung von Bit eins, des zweiten Bit von rechts. Ist der geladene Wert Null, so zeigt das Zero-Flag eine Eins, im anderen Falle eine Null. Mit den Branchbefehlen BEQ und BNE kann daraufhin entsprechend verzweigt werden. Branch Equal ist gleichbedeutend mit Branch Zero und heißt soviel wie: „Verzweige, wenn Zero-Flag gleich eins".
Branch Not Equal verzweigt im umgekehrten Falle.

Das Vorzeichenflag

Datensätze in Standard-ASCII, wobei nur Werte bis #$7F vorkommen, lassen sich sogar noch kürzer abspeichern. Ein extra Byte für die Endemarkierung kann entfallen. Es braucht nur Bit sieben des letzten Zeichens auf Eins gesetzt werden. Das Vorzeichenflag zeigt dieses an.

```
LDA #$01
LDA #$81
```

Bit sieben, das ganz links wiedergegeben wird, ist das Vorzeichenflag. Es heißt so, da beim Rechnen mit negativen Zahlen definiert wird, daß Zahlen von #$00 bis #$7F positiv seien und Zahlen von #$80 bis #$FF negativ. Bit sieben des Statusregisters gibt also Bit sieben eines geladenen oder auf Grund einer sonstigen Operation erhaltenen Wert eines Registers oder Speicherinhaltes wieder.
Bit sieben läßt sich ganz einfach mit dem Vorzeichenflag abfragen. Der Prozessor sollte aber auch die restlichen Bit aus einem Register oder aus einer Speicherzelle abfragen können. Einen Befehl, mit dem wir dieses leicht bewerkstelligen können, kennen wir bereits.

```
LDA #$01
AND $0001
LDA #$02
AND $0001
LDA #$04
AND $0001
LDA #$08
AND $0001
LDA #$10
AND $0001
LDA #$20
AND $0001
LDA #$40
AND $0001
LDA #$80
AND $0001
```

Wir setzen im Akku alle Bit auf Null, mit Ausnahme desjenigen, das wir abfragen wollen. Durch den AND-Befehl werden alle Bit, die uns nicht interessieren, zu Null. An der Stelle, die wir mit einer Eins maskiert hatten, gelangt das gewünschte Bit aus der Speicherstelle in den Akku. Wenn dieses Null war, so ist der gesamte Akku Null und das Zero-Flag ist gesetzt. Im anderen Falle wissen wir, daß uns eine Eins vorliegt.
Der *AND*-Befehl ändert auch den Akku-Inhalt. Manchmal wäre es wünschenswert, wenn dieser uns erhalten bliebe. Außerdem kann mit dem AND-Befehl nur ein einziges Bit über das Zero-Flag abgefragt werden. Ein paar Möglichkeiten mehr bietet der Bit-Befehl. Um Bit sieben abzufragen, braucht nicht extra der Wert eines Registers durch Hereinladen zerstört werden. Zusätzlich zu Bit sieben im Vorzeichenflag wird Bit sechs im Überlauf-Flag erfaßt. Die restlichen Bit können auf dieselbe Art, wie wir es bereits vom AND-Befehl her kennen, abgefragt werden.
Die Besonderheit des *BIT*-Befehls ist, daß zwar auch eine Undierung stattfindet, jedoch das Ergebnis nur im Setzen des Zero-Flags sichtbar wird, und so der Akkuinhalt unverändert bleibt.

```
LDA #$01
BIT $0001
LDA #$02
BIT $0001
LDA #$04
BIT $0001
LDA #$08
BIT $0001
LDA #$10
BIT $0001
LDA #$20
BIT $0001
```

Wir haben die Abfragen jetzt nur bis Bit fünf ausgeführt, da das Statusregister uns sowieso immer die Werte von Bit sechs und sieben wiedergab. Da das Überlauf-Flag auf Eins steht, gibt uns dieses die Gelegenheit, den Befehl *CLV* auszuprobieren. Es funktioniert wie gewünscht.
Wir können den Wert #$00 abfragen, wir können abfragen, ob ein Wert größer als #$7F ist, sogar vor einzelnen Bit brauchen wir nicht zu kapitulieren. Dennoch fehlt noch irgend etwas. Was tun wir denn, wenn Datensätze nicht mit einer Null abgeschlossen sind und auch nicht mit einem Wert größer als #$7F, sondern schlicht mit einem Carriage-Return oder einem sonstigen Wert?

Wir müßten demnach vergleichen können, ob wir einen bestimmten Wert vor uns haben, oder nicht. Dafür gibt es den *Compare*-Befehl, der zwei Werte miteinander vergleicht. Vergleichen können wir die Registerinhalte unmittelbar mit einem bestimmten Wert, oder aber mit dem Inhalt einer beliebigen Speicherzelle. Die Compare-Befehle sind:

```
CMP
CPX
CPY
```

CMP vergleicht mit dem Akkuinhalt, die beiden anderen Befehle sind wohl nicht schwer zu erraten.

```
LDA #$04
CMP #$03
```

Das Zero-Flag zeigt eine Null.

```
LDA #$04
CMP #$04
```

Das Zero-Flag zeigt eine Eins zum Zeichen, daß die beiden verglichenen Werte übereinstimmen. Einen Vergleich können wir uns so vorstellen, daß vom Wert im Register der verglichene Wert subtrahiert wird, ohne daß der Wert im Register verändert wird. Wie beim Bit-Befehl erscheint das Resultat nur im Statusregister.
Die Namensgebung der Befehle BEQ und BNE ist auf diesen Vergleich zurückzuführen. Stimmen die beiden Werte überein, so ist die Differnz der Werte Null, das Zero-Flag somit gesetzt und BEQ verzweigt entsprechend. *Branch Equal* heißt somit: „Verzweige, wenn Gleichheit besteht."
Von BASIC her kennen wir:

```
IF A=B
IF A<>B
IF A<B
IF A>B
```

Die ersten beiden Fälle sind in Maschinensprache bereits abgehandelt. Die letzten beiden fehlen uns noch. Wir beobachten bei den folgenden Vergleichen einmal das Carry-Flag, ob hier ein Unterschied wahrnehmbar ist.

```
LDA #$04
CMP #$03
CMP #$05
CMP #$04
```

Ist der Registerinhalt größer oder gleich dem zu vergleichenden Wert, so wird das Carry gesetzt, im anderen Falle gelöscht.

Mit den Branch-Befehlen *BCS*, Branch Carry SET, und *BCC*, Branch Carry Clear, können wir nach Wunsch verzweigen. Beim *CMP*-Befehl wird auch noch das Vorzeichenflag beeinflußt. Eine große Bedeutung kommt diesem Umstand wohl nicht zu. Wir können aber leicht noch einmal nachprüfen, ob Bit sieben in einem Register gesetzt ist, wenn zwischendurch auch schon andere Ladeoperationen erfolgt sein sollten.

```
LDA #$90
LDX #$01
CMP #$00
```

Wir besitzen jetzt ein fundiertes Wissen über die Programmlogik unseres Rechners. Ob wir unser Gerät Rechner nennen dürfen, werden uns die Rechenbefehle zeigen.

VI. DIE RECHENOPERATIONEN DER CPU

Addition, Subtraktion, Rotations-, und Schiebe-, Ikrementier- und Dekrementierbefehle sind Operationen, mit denen der Prozessor aufwartet. Multiplikationen und höhere Rechenoperationen lassen sich mit diesen Befehlen realisieren.

Ihr Taschenrechner wartet mit Grundrechenarten, Wurzeln, Potenzen und vielen mathematischen Funktionen auf. BASIC besitzt ebenso eine Menge von Rechenoperationen. Der Rechner muß also eine Menge können. Mancher wird wohl sehr verblüfft sein, wie wenig die CPU tatsächlich nur kann. Sie wartet nicht mit vielstelligen Dezimalzahlen und Exponenten, sondern nur mit Byte-weiser Addition und Subtraktion auf.
Multiplikation und Division sind für sie bereits Fremdwörter. Solche Wörter wie Sinus oder Cosinus getrauen wir uns gar nicht mehr in den Mund zu nehmen. Wenn auch der arithmetische Befehlssatz nicht sehr umfangreich ist, so läßt sich doch mit diesem alles programmieren, was das Herz begehrt.

Addition und Subtraktion

Wir haben die Wahl zwischen binärem oder BCD-System. Vor dem Addieren ist das Carry-Flag auf Null zu setzen, vor das Subtrahieren auf Eins.

```
CLC
LDA #$04
ADC #$03
```

Wie gewünscht, ist das Ergebnis im Akku sichtbar. Wir nehmen nun Zahlen, die geringfügig größer sind.

```
CLC
LDA #$08
ADC #$07
```

Die dezimale Darstellung zeigt den Wert 15, wie es sein soll. Die hexadezimale Darstellung wartet mit #$0F auf, was zwar noch vorstellbar ist, jedoch für uns Zehn-Finger-Menschen schon etwas unanschaulich wirkt. Doch es kommt noch schlimmer.

```
CLC
LDA #$28
ADC #$49
```

Weder die deziamel Darstellung noch die Hex-Darstellung zeigen an, daß 28 und 49 die Summe 77 ergeben müßten. Wir und die CPU benützen eben zwei unterschiedliche Zahlensysteme. Wenn wir die Zahlen #$28 und #$49 in die Index-Register laden, sehen wir, daß #$28 eben nicht auch dezimal 28 bedeutet. Die Summe der Werte in den Indexregistern ergibt jedoch genau den Wert, den wir im Akku vorfinden. Wir brauchen nur die dezimal dargestellten Werte miteinander vergleichen. Keine Sorge, wir brauchen uns deshalb nicht umgewöhnen, von nun an in Hex-Zahlen zu rechnen.

HEX, DEZIMAL UND BCD

Die Schwierigkeit ist nur, daß der Rechner und wir andere Zahlensysteme verwenden. Der Programmanwender merkt davon nichts.
Der Programmierer hat dafür Sorge zu tragen, daß die dezimal eingegebenen Zahlen in Hex-Zahlen umgewandelt werden. Die CPU kann nun auf ihre Weise mit diesen Zahlen rechnen. Das Ergebnis muß wieder in das Dezimalsystem zurückübersetzt werden. Wenn der Rechner allerdings nicht einmal multiplizieren kann, so ist dieses wohl gerade kein leichtes Unterfangen. Glücklicherweise kommt uns das Statusregister mit dem Dezimalflag einen Schritt entgegen, so daß wir die Rechenleistungen gut verfolgen können.

```
SED
CLC
LDA #$08
ADC #$07
LDA #$28
ADC #$49
```

Bei der Subtraktion setzen wir das Carry-Flag.

```
SEC
LDA #$77
SBC #$49
```

Das Ergebnis entspricht auch unseren Erwartungen. Wenn der Rechner nur mehr Stellen zulassen würde! Mit dem Carry-Flag hat es noch eine besondere Bewandtnis.

```
CLC
LDA #$60
ADC #$50
```

Wir beobachten, daß das Carry-Flag nun gesetzt ist. Da die Summe aus 50 und 60 größer als 99 ist, ist ein Übertrag entstanden, der bei weiteren Additionen dazuaddiert wird.

```
TAX
LDA #$01
ADC #$02
```

Eins und zwei sind plötzlich vier, da der Übertrag aus der vorangegangenen Rechnung Berücksichtigung fand.
Betrachten wir die beiden Rechnungen im Zusammenhang, so haben wir eigentlich die Summe aus 160 und 250 gebildet. Akku und X-Register enthalten 0410. Wenn wir mehrmals hintereinander die einzelnen Stellen einer vielstelligen Zahl unter Berücksichtigung der Überträge addieren, so sind uns von der Stellenzahl her keine Grenzen gesetzt. Wir brauchen nur einen bestimmten Speicherbereich unserer Rechnenoperation vorsehen. Die jeweilige Summe aus den beiden Summanden schreiben wir wieder in den Speicher.

```
BRK
M3000
F3000,3FFF,0
M3000
```

Nach der Adreßangabe >3000 finden wir die Hexzahlen 00 00 00. Diese ändern wir in eine beliebige sechsstellige BCD-Zahl. Mit der Zeile weiter unten wollen wir genauso verfahren.

```
G
CLC
LDA $3002
ADC $300A
STA $3012
LDA $3001
ADC $3009
STA $3011
LDA $3000
ADC $3008
STA $3010
BRK
M3000
```

Wir können nun das Ergebnis betrachten. Wenn nicht vorne bei den Hunderttausendern noch ein Überlauf geschehen ist, sollte unser Ergebnis stimmen.
Für mehrstellige Subtraktionen dient ebenso wieder das Carry-Flag. Normalerweise müßte es in diesem Falle Borrow-Flag heißen. Für unsere CPU gilt, daß ein gesetztes Carry ein gelöschtes Borrow bedeutet und umgekehrt. Andere CPUs können auch eine andere Regelung besitzen. Wir versuchen uns nun an der mehrstelligen Subtraktion. Zahlen brauchen wir keine hierzu noch eingeben. Wir löschen lediglich eine Zeile im Monitor.

```
F3000,3007,0
M3000
```

Der erste Summand aus unserer Addition ist verschwunden. Von der früheren Summe ziehen wir den zweiten Summanden ab und sollten wieder die erste Zeile, wie gehabt, bekommen.

```
G
SEC
LDA$3012
SBC $300A
STA $3002
LDA$3011
SBC $3009
STA $3001
LDA$3010
SBC $3008
STA $3000
BRK
M3000
1;
```

Für unsere Zwecke reicht das Rechnen mit positiven Zahlen vollständig. Die CPU kennt schließlich nur positive Adressen von $0000 bis $FFFF. Wenn einmal etwas errechnet werden muß, wie die Länge eines Speicherbereiches aus der Anfangsadresse und der Endadresse, so ist maximal eine zweistellige Rechnung vonnöten. Für diejenigen, die sich einmal vornehmen sollten, eine Tabellenkalkulation oder ein Buchrechnungsprogramm in Maschinensprache zu programmieren, sind auch das Vorzeichenflag und das Überlauf-Flag von Bedeutung. Wir definieren uns nun, daß Zahlen von #$00 bis #$7F positiv sein sollen und Zahlen von #$80 bis #$FF negativ, und schenken den Status-Bit sechs und sieben nun unsere Aufmerksamkeit. Zuerst verlassen wir aber erst einmal den Monitor und setzen das Dezimalflag zurück.

```
G
CLD
```

Wir können jetzt mit dem Rechnen beginnen.

```
CLC
LDA #$40
ADC #$50
```

Das Ergebnis #$90 ist als negative Zahl definiert, was das Vorzeichen-Flag auch angibt. Nur kann die Summe aus zwei positiven Zahlen wohl kaum negativ sein. Daher ist das Überlauf-Flag gesetzt. Es signalisiert, daß das Vorzeichen-Flag den Sachverhalt nicht richtig wiedergibt.

```
CLC
LDA #$40
ADC #$20
```

Beide Flags sind null. Die Zahl ist positiv. Das Vorzeichen-Flag zeigt es richtig an.

```
CLC
LDA #$40
ADC #$FF
```

Die Hexzahl #$FF entspricht einer negativen Eins. Das Ergebnis ist daher positiv und richtig.

```
SEC
LDA #$10
SBC #$20
```

Das Ergebnis ist negativ, das Vorzeichen-Flag zeigt dieses an.

```
SEC
LDA #$90
SBC #$20
```

Von einer negativen Zahl wird eine positive abgezogen. Das Ergebnis muß daher negativ sein. Daß das Vorzeichen-Flag den Sachverhalt nicht richtig wiedergibt, wird vom Überlauf-Flag signalisiert.

```
SEC
LDA #$05
SBC #$90
```

Von einer positiven Zahl wird eine negative subtrahiert. Das Ergebnis muß daher positiv sein. Das Vorzeichen-Flag gibt es richtig wieder.

```
SEC
LDA #$30
SBC #$90
```

Das Ergebnis muß aus den vorher genannten Gründen wieder positiv sein. Das Überlauf-Flag widerspricht dem Vorzeichen-Flag.

```
CLC
LDA #$90
ADC #$90
```

Zu einer negativen Zahl wird eine negative Zahl addiert. Das Ergebnis muß negativ sein. Das Überlauf-Flag sagt aus, daß das Vorzeichen-Flag nicht recht hat.
Wir sprachen davon, daß die CPU nicht multiplizieren könne. Ganz richtig ist dieses jedoch nicht. Wir, die im Dezimalsystem rechnen, benötigen das Einmaleins bis zum Neunereinmaleins. Das Multiplizieren mit zehn geschieht durch einfaches Linksverschieben. Im Binärsystem, das nur die Zahlen null oder eins kennt, brauchen wir nur das Einereinmaleins und das Verschieben, das einer Multiplikation mit zwei gleichkommt. Mit eins multiplizieren ist keine Kunst, da dieses sich von selbst erübrigt. Für das Verschieben stehen uns diverse Befehle zur Verfügung.

```
LDA #$01
ASL
```

Der Akkumulatorinhalt wird durch den *ASL-Befehl,* Arithmetic Shift Left, um eins nach links geschoben. Wiederholen Sie den ASL-Befehl, bis die Eins aus dem Akku verschwindet. Ist dieses der Fall, erscheint ein gesetztes Bit im Carry-Flag. Nach einem weiteren Befehl ist auch dieses verschwunden. Sie brauchen den ASL-Befehl nicht jedesmal neu einzugeben. Unser Programm hat eine Wiederholautomatik. Wenn Sie nur die Return-Taste drücken, wird der zuletzt eingegebene Befehl immer wieder ausgeführt.

```
LDA #$01
ROL
```

Fas das gleiche wie der ASL-Befehl scheint der *ROL-Befehl,* **RO**tate Left, zu bewirken, daß dieses jedoch nicht so ist, sehen wir, wenn die Eins aus dem Akku verschwunden und im Carry-Flag gelandet ist. Beim nächsten ROL-Befehl geht sie von dort wieder in den Akku über.

```
LDA #$80
LSR
```

Logical Shift Right bewirkt dasselbe wie ASL, nur in umgekehrter Richtung.

```
LDA #$02
ROR
ROR
ROR
```

Rotate Right funktioniert genau wie ROL in umgekehrter Richtung. Diese Befehle greifen nicht nur auf den Akku zu, sondern es lassen sich auch beliebige Speicherstellen dadurch verändern. Auf ASL und LSR könnte zur Not auch verzichtet werden, da durch Löschen des Carry-Flags dafür gesorgt werden kann, daß die Rotate-Befehle stets eine Null in den Akku oder die Speicherstelle rotieren.
Das Hereinnehmen des Carry-Flags durch die Rotate-Befehle ist äußerst nützlich. Denken wir hierzu nur einmal an eine Grafik-Hardcopy. Durch acht mal acht Bit wird das Bit-Muster eines Zeichens bestimmt. Der Bildschirm hat es zu acht Reihen mit jeweils einem Byte, der Drucker jedoch braucht es in Spaltenform. Nachdem die Matrix aus dem Bildschirmbereich herauskopiert wurde – schließlich wollen wir den Bildschirminhalt nicht zerstören –, können wir mit ROL-Befehlen für den Drucker zurechtschneidern. Aus jedem der acht Bildschirm-Byte rotieren wir je ein Bit heraus und rotieren es mit einem zweiten ROL-Befehl in den Akku. Nachdem der Akkuinhalt ausgegeben ist, holen wir uns nach demselben Verfahren die zweite Spalte heraus, bis alle acht Spalten ausgegeben sind.
Zum Multiplizieren laden wir jeweils eine Zahl, deren Produkt nicht größer als 255 oder #$FF betragen soll, in die Indexregister. Das Produkt soll nach der Multiplikation im Akku stehen, die Faktoren noch in den Indexregistern.

LDX #$06
LDY #$09

Es gilt nun ein geeignetes Verfahren, die Multiplikation zu finden. Wir laden hierzu den Akku mit Null und legen die beiden Werte in zwei Speicherstellen ab, so daß wir sowohl schieben als auch addieren können.

LDA #$00
STX $3000
STY $3001

Eine Zahl können wir dadurch mit sechs multiplizieren, daß wir sie in den Akku laden. Dadurch haben wir sie einmal erfaßt. Indem wir sie nach links verschieben, haben wir sie bereits mit zwei multipliziert. Wenn wir sie noch einmal dazuaddieren, ist sie schon mit drei malgenommen. Ein abschließendes Linksverschieben bringt das gewünschte Ergebnis.
Für beliebige Zahlen, die bis zu acht Bit lang sein dürfen, benötigen wir noch ein paar weitere Befehle. Acht Bit sind zu untersuchen. Daher muß unser Rechner bis acht zählen können. Zum Zählen dienen Befehle, mit denen vorwärts und rückwärts gezählt werden kann.

LDY #$08
DEY

Bei jedem DEY wird das Y-Register um Eins erniedrigt, bis beim Erreichen von Null das Zero-Flag diesen Zustand anzeigt. Beim weiteren Decrementieren spricht das Vorzeichen-Flag an.

INY

Mit INY kann das Y-Register hochgezählt werden. Die restlichen Befehle sollen nun nicht gleich ausprobiert werden, da wir den Inhalt des X-Registers weder decrementieren noch incrementieren wollen, da wir diesen, bevor wir nicht multipliziert haben, gerne unverändert hätten.
Neben den Indexregistern können auch Speicherstellen angesprochen werden. Es folgt eine Auflistung der Increment- und Decrement-Befehle

INX
DEX
INY
DEY
INC
DEC

Wir laden also das Y-Register mit dem Wert #$08 und unser Algorithmus kann beginnen. Da hierbei auch Verzweigungen stattfinden, sei er in einzelnen Schritten verfaßt.

1. **ASL**
2. **ASL $3000**
3. **Wenn Carry = 0, dann Sprung nach 6.**
4. **CLC**
5. **ADC $3001**
6. **DEY**
7. **Wenn Zero = 0, dann Sprung nach 1.**
8. **LDY $3001**

Etwas mühsam ist es, solch einen Algorithmus mit der Hand durchzuprobieren. Zum Programmieren fehlt uns fast nichts mehr, denn unser Befehlssatz ist bereits nahezu vollständig. Uns fehlt nur noch der Befehl, um ein Unterprogramm anzuspringen, und der Befehl für den Rücksprung. Der Befehl für den Rücksprung heißt *RTS*. Wir können ihn in unserem CPU-Programm nicht aufrufen. Genauso, wie die Branch-Befehle verwenden wir ihn erst in einem richtigen Programm.
Der *JSR*-Befehl, der ein Unterprogramm anspringt, ist auch schon in unserem CPU-Programm verfügbar. Wir dürfen ihn allerdings erst verwenden, wenn wir ein anzuspringendes Maschinenprogramm vor uns haben. Besonders beachtet werden muß bei diesen Befehlen, daß der JSR, Jump SubRoutine, die Rücksprungadresse auf dem Stapel ablegt. Nur so kann *RTS*, ReTurn from Subroutine, wieder an die richtige Stelle im Hauptprogramm zurückkehren. Dies bedeutet, daß ein auf den Stapel gepushter Wert nicht nach einem Sprung in eine Unterroutine so ohne weiteres abgehoben werden kann. Anstatt dem geretteten Akkuinhalt würde ein Pull-Befehl ein Byte der Rücksprungadresse abheben.
Der RTS hätte als Rücksprungadresse eine Kombination von beretteten Akkuinhalt als High-Byte und High-Byte-Rücksprungadresse als Low-Byte vor sich und würde ganz schön ins Nirwana stürzen. Außer den zwei soeben genannten Befehlen gibt es noch einen unbedingten Sprung namens *JMP*, nur für Sprünge in einem Bereich von $7F Adressen vorwärts und $80 Adressen rückwärts eingesetzt werden kann, sondern den Programmzähler auf jede beliebige Speicheradresse einstellen kann. Stapelbeeinflussungen finden hierbei nicht statt.
Um vollständig zu sein, wollen wir auch einen ganz unscheinbaren Befehl nicht unerwähnt lassen. Sie können ihn ruhig gleich ausprobieren. Er heißt *NOP*, was No OPeration bedeutet. Dieser Befehl bewirkt nichts. Er wird gerne als Platzhalter verwendet, so wie wir es auch im Maschinenteil unseres CPU-Programmes taten. An eine bestimmte Stelle des Programms setzen wir je nach auszuführendem Befehl entweder ein zwei oder drei Byte lange Befehle ein. Damit die CPU bei ein oder zwei Byte langen Befehlen nicht über den zurückgebliebenen Rest der drei Byte langen Befehle stolpert, füllten wir die verbleibende ein oder zwei Adressen mit NOP-Codes auf. Wir schreiben jetzt unser erstes Maschinenprogramm. Da wir den Algorithmus für die Multiplikation bereits notiert hatten, fällt es uns bestimmt nicht schwer, diesen mit Hilfe des Monitors einzugeben.

```
BRK
A3002 LDA #$00
STX $3000
STY $3001
LDY #$08
ASL
ASL $3000
BCC $3010
CLC
ADC $3001
DEY
BNE $300C
LDY $3001
RTS
```

Der Sprungbefehl bei Adresse $3010 stimmt noch nicht ganz. Wir haben, da wir die Sprungadresse beim Niederschreiben noch nicht gekannt haben, einen provisorischen Wert eingesetzt. Da jetzt die Adresse des DEY-Befehles bekannt ist, ersetzen wir BCC $3010 durch BCC $3016. Den Monitorbefehl zum Betrachten eines Speicherbereichs im Hexformat kennen wir bereits. Zum Betrachten eines Maschinenprogramms wäre es sehr schön, wenn dieses sich auch in Assembler-Mnemonics anschauen ließe. Auch dafür gibt es einen Monitorbefehl. Er heißt *Disassemble* und wird mit dem Buchstaben **D**, gefolgt von einer Adresse eingegeben. Sie dürfen ruhig einmal den Bildschirm löschen. Danach sehen wir uns das Maschinenprogramm wieder an.

```
D3002
```

Ganz wird das Multiplikationsprogramm durch diesen Monitorbefehl nicht angezeigt. Um den Rest zu disassemblieren, ist nur noch ein D ohne nachfolgende Adresse einzugeben.

```
D
```

Disassembliert sollte das Programm sich nun darstellen, wie es nachfolgend zu sehen ist.

```
. 3002  a9 00 30  lda  #$00
. 3004  8e 00 30  stx  $3000
. 3007  8c 01 30  sty  $3001
. 300a  a0 08     ldy  #$08
. 300c  0a        asl
. 300d  0e 00 30  asl  $3000
. 3010  90 04     bcc  $3016
. 3012  18        clc
. 3013  6d 01 30  adc  $3001
. 3016  88        dey
. 3017  d0 f3     bne  $300c
. 3019  ac 01 30  ldy  $3001
. 301c  60        rts
```

Rechts neben dem Punkt erscheint die jeweilige Speicheradresse. Die nachfolgenden ein, zwei oder drei Byte sind der Operationscode und ein oder zwei Parameter. Darauf folgen die disassemblierten Mnemonics. Interessant ist, daß bei der absoluten Adressierung dem Operationscode erst das Low-Byte der Speicheradresse und dann erst das High-Byte folgt. Den Operationscodes für die bedingten Sprünge folgt nicht eine absolute Sprungadresse, sondern nur ein einziges Byte, das angibt, um wieviel Adressen der Programmzähler verstellt werden soll. Im Verzweigungsfalle landet der Programmzähler beim BCC in Adresse $3010 nicht bei $3012 sondern, vier Adressen weiter, bei $3016. Zahlen ab $80 sind negativ, wobei $ff der Dezimalzahl −1 entspricht und $80 der Dezimalzahl −128.
Der Sprungbefehl in Adresse S3017 landet daher bei Adresse $300 c.
Relative Sprünge bringen einen entscheidenden Vorteil. Die Programme sind verschiebbar. Wir kopieren unser Maschinenprogramm.

```
T3002,301C,3100
D3100
```

Die disassemblierten Sprungadressen sind andere, als wir ursprünglich erfaßt hatten. Der Disassembler zeigt jetzt automatisch die richtigen neuen Sprungadressen an, da ja nur die Sprungweite im Maschinenprogramm steht.

```
G
```

Wir stoßen auf eine Eigenheit des Maschinenmonitors. Sobald wir Assemblierungen vorgenommen haben, landen wir beim Fortsetzen mit *Go* nicht, wie wir es gewohnt sind, im CPU-Programm, sondern im Eingabemodus des BASIC-Interpreters. Unser CPU-Programm ist nicht verlorengegangen, es läßt sich wieder starten.

```
RUN
```

Jetzt haben wir die Gelegenheit, den JSR-Befehl und unser Maschinenprogramm auszuprobieren.

```
LDX #$05
LDY #$07
JSR  $3002
```

Das Multiplizieren klappt ja direkt hervorragend. Mit dem Befehlssatz des Prozessors sind wir am Ende angelangt. Zu besprechen sind lediglich noch die verschiedenen Adressierungsarten.

VII. ADRESSIERUNGSARTEN DER CPU

Neben impliziter, unmittelbarer und absoluter Adressierung bietet die CPU unseres Rechners eine Vielzahl weiterer Adressierungen. Dreizehn sind es insgesamt. Mit indirekter und indizierter Adressierung können wir ganze Speicherbereiche durchwühlen. Eine Befehlstabelle gibt uns den Überblick.

In der Beschreibung der Adressierungsarten sind auch Befehlslänge und Befehlsdauer angegeben. Die Befehlsdauer mißt sich in Taktzyklen. Im Normalfall wird unsere CPU mit der Frequenz von einem Megahertz getaktet. Das sind eine Million Takte in der Sekunde. Ein Takt dauert so eine Millionstel Sekunde, anders ausgedrückt, eine Mikrosekunde. Ein Befehl, der in zwei Taktzyklen abgearbeitet wird, braucht somit zwei Mikrosekunden. 500000 solcher Befehle könnten also in der Sekunde durchgeführt werden, eine ganz erkleckliche Anzahl. Kein Wunder also, daß in Maschinensprache geschriebene Programme als ungeheuer schnell gelten.
Manche Adressierungsarten arbeiten schneller als andere. Manche nehmen weniger Programmspeicher in Anspruch, weil die Befehlslänge eventuell kürzer ist. Andere Adressierungsarten fixieren uns nicht auf bestimmte Adressen, sondern erlauben uns, Routinen zu schreiben, die auf jeden beliebigen Speicherbereich zugreifen können. In der Befehlstabelle sehen wir, welche Adressierungen für welche Operationen zur Verfügung stehen, und können so das für unseren Zweck geeignetste auswählen.

A. Unmittelbare Adressierung

Beispiel: LDY #$05
Befehlslänge: Zwei Byte
Taktzyklen: Zwei

Dem Operationscode folgt unmittelbar das Daten-Byte, das mit dem Akkumulatorinhalt verknüpft wird.

B. Absolute Adressierung

Beispiel: STA $3000
Befehlslänge: Drei Byte
Taktzyklen: Drei bis sechs

Dem Operationscode folgt eine Zwei-Byte-Adresse. Hierbei wird erst das Low- und dann das High-Byte im Speicher abgelegt. Der JMP-Befehl benötigt drei Taktzyklen, der JSR sechs. Befehle, die eine Änderung einer Speicherzelle bewirken, benötigen sechs Taktzyklen. Die Ausnahme bildet der Store-Befehl mit vier Taktzyklen. Lade- und sonstige Befehle, die den Inhalt einer Speicherzelle mit einem CPU-Register verknüpfen, dauern ebenfalls vier Taktzyklen.

C. Zero-Page-Adressierung

Beispiel: LDA $01
Befehlslänge: Zwei Byte
Taktzyklen: Drei bis fünf

Um Adressen von $00 bis $FF anzusprechen, kann auf ein High-Adress-

Byte verzichtet werden. Dem Operationscode folgt nur ein einziges Adress-Byte. Kürzer ist nicht nur die Länge des Befehls, sondern auch die Abarbeitungszeit. Ein Taktzyklus wird gespart.

D. Akku-Adressierung

Beispiel: ASL
Befehlslänge: Ein Byte
Taktzyklen: Zwei

Da keine Werte übergeben werden und keine Speicherzelle zu adressieren ist, wird nur der Operationscode benötigt.

E. Implizite Adressierung

Beispiel: SEC
Befehlslänge: Ein Byte
Taktzyklen: Zwei bis sechs

Die Adressierung ist bereits durch den Operationscode bestimmt. Daten oder Adreßparameter erübrigen sich daher. Befehle, die nicht auf den Speicher zugreifen, dauern zwei Taktzyklen. Push-Befehle vier, die Rücksprünge RTS und RTI benötigen gar sechs Zyklen. Mit RTI wird von einem Interrupt wieder zurückgesprungen. Da ein Interrupt nicht nur die Rücksprungadresse, sondern auch das Statusflag auf den Stapel rettet, wird dieses durch RTI wieder restauriert.

F. Indiziert-indirekte Adressierung

Beispiel: LDA ($D0,X)
Befehlslänge: Zwei Byte
Taktzyklen: Sechs

Dem Operationscode folgt ein Adreß-Byte. Indirekt adressiert heißt, daß nicht die durch den Adreßparameter bestimmte Zelle adressiert wird, sondern, daß in der spezifizierten und der nachfolgenden Speicherstelle sich erst die Adresse der Zelle findet, auf die die Operation zugreift. Indiziert heißt, daß zur Adreßbildung zusätzlich ein Index-Register einbezogen wird. Bei der indiziert-indirekten Adressierung der CPU 6502 ist dieses das X-Register.
Der Inhalt des X-Registers wird in unserem Beispiel zur Adresse $D0 addiert. Wir erhalten so die Adresse der Speicherstelle. In dieser und der nachfolgenden finden wir endlich die Adresse unserer Speicherstelle, die es anzusprechen gilt. Bei der Summierung der X-Register und der Zero-Page-Adresse bleibt ein Übertrag unberücksichtigt, so daß für indirekte Adressierungen auch wirklich nur Speicherstellen von $00 bis $FF in Frage kommen.

Die meisten Zero-Page-Adressen sind bereits für Operationen des Betriebssystems vergeben. Zur freien Verfügung steht dem C16/116/Plus4-User der Bereich von $D0 bis $E8, in dem er nach Belieben schalten und walten kann. Um die Sache noch etwas zu verdeutlichen, nehmen wir einmal an, das X-Register hätte den Wert #$02, in der Speicherstelle $D2 stünde der Wert #$00 und in $D3 der Wert #$30. Die adressierte Speicherstelle wäre in diesem Falle $3000.

G. Indirekt-indizierte Adressierung

Beispiel: LDA $(D0),Y
Befehlslänge: Zwei Byte
Taktzyklen: Fünf bis sechs

Der *Store*-Befehl benötigt sechs Zyklen, die anderen Befehle nur fünf. Diese Adressierungsart wird öfter benötigt als die indiziert-indirekte. Zur Adresse in den zwei Speicherstellen in der Zero-Page wird der Inhalt des Y-Registers addiert. Die Summe davon ist die Adresse unserer anzusprechenden Speicherzelle. Hätte das Y-Register den Wert #$81, die Speicherstelle $D0 den Wert #$80 und $D1 den Wert #$30, so würde durch die Operation die Speicherstelle $3101 angesprochen werden. Überträge bei der Summierung der Adress-Low-Byte werden berücksichtigt. Bei allen indizierten Adressierungsarten ist, wenn solche Pageüberschreitungen auftreten, ein Taktzyklus hinzuzuaddieren.

H. Zero-Page-indizierte Adressierung mit Indexregister X

Beispiel: LDA $D0,X
Befehlslänge: Zwei Byte
Taktzyklen: Vier bis sechs

Die Abarbeitungszeit der Befehle entspricht derjenigen der absoluten Adressierung. Zur Adresse wird der Inhalt des X-Registers addiert. Es können nur Speicherstellen von $00 bis $FF angesprochen werden. Ein Übertrag bei der Addition der Adressen wird unterschlagen.

I. Direkt indizierte Adressierung mit Indexregister X

Beispiel: LDA $3000,X
Befehlslänge: Drei Byte
Taktzyklen: Vier bis sieben

Der *STA*-Befehl benötigt fünf Taktzyklen, sonstige Befehle, die den Inhalt einer Speicherstelle verändern, gar sieben. Alle übrigen Operationen geben sich mit vier Zyklen zufrieden.

Wie bei der direkten Adressierung besteht die Basisadresse aus zwei Byte.
Anders als bei der Zero-Page-indizierten Adressierung wird bei der Bildung der Operandenadresse ein aus der Summierung der Low-Byte resultierender Übertrag berücksichtigt. Hätte X den Wert #$81, so würde mit LDA $3080,X die Speicherstelle $3101 angesprochen.

J. Direkt indizierte Adressierung mit Indexregister Y

Beispiel: LDA $3000,Y
Befehlslänge: Drei Byte
Taktzyklen: Vier bis fünf

Zur Basisadresse wird das Y-Register addiert. Der STA-Befehl braucht fünf Taktzyklen, die anderen Befehle nur vier.

K. Relative Adressierung

Beispiel: BCC $3016
Befehlslänge: Zwei Byte
Taktzyklen: Zwei bis drei

Dem Operanden folgt ein Byte, das die Sprungweite signalisiert. Zahlen von #$80 bis #$FF gelten als negativ. Sprünge können so nur in einem Bereich von Null bis 127 Adressen nach vorne und –1 bis 128 Adressen nach rückwärts erfolgen. Ist die Verzweigungsbedingung nicht erfüllt, so benötigt der Befehl zwei Taktzyklen, im Verzweigungsfalle drei. Bei Page-Überschreitung ist ein weiterer Taktzyklus hinzuzuaddieren.

L. Indirekte Adressierung

Beispiel: JMP ($0324)
Befehlslänge: Drei Byte
Taktzyklen: Fünf

Eine indirekte Adressierung, die nicht auf Zero-Page-Adressen beschränkt ist, kennt nur der *JMP*-Befehl. In unserem Falle erfolgt der Sprung nach der Adresse, die in den Speicherstellen $0324 und $0325 vermerkt ist. Einige wichtige Betriebssystemroutinen unserer Commodore-Rechner werden indirekt über sogenannten Sprungvektoren aufgerufen, die im RAM lokalisiert sind. Durch das Ersetzen der Adresse in den Speicherstellen $0324 und $0325, ist es ein Leichtes, eine neue Ausgaberoutine zu schreiben, die zum Beispiel für einen Epson-Drucker den Commodore-ASCII in Standard-ASCII umwandelt. Die Adresse eines direkten Sprunges im ROM hätten wir nicht manipulieren können, da aus dem ROM ja bekanntlich nur gelesen werden kann.

	A	B	C	C	E	F	G	H	I	J	K	L	M	
ADC	*	*	*			*	*	*	*	*				NV--ZC
AND	*	*	*			*	*	*	*	*				N---Z-
ASL		*	*	*				*	*					N---ZC
BCC											*			------
BCS											*			------
BEQ											*			------
BIT		*	*											NV--Z-
BMI											*			------
BNE											*			------
BPL											*			------
BRK					*									------
BVC											*			------
BVS											*			------
CLC					*									-----C
CLD					*									--D---
CLI					*									---I--
CLV					*									-V----
CMP	*	*	*			*	*	*	*	*				N---ZC
CPX	*	*	*											N---ZC
CPY	*	*	*											N---ZC
DEC		*	*					*	*					N---Z-
DEX					*									N---Z-
DEY					*									N---Z-
EOR	*	*	*			*	*	*	*	*				N---Z-
INC		*	*					*	*					N---Z-
INX					*									N---Z-
INY					*									N---Z-
JMP		*										*		------
JSR		*												------
LDA	*	*	*			*	*	*	*	*				N---Z-
LDX	*	*	*							*			*	N---Z-
LDY	*	*	*					*	*					N---Z-
LSR		*	*	*				*	*					N---ZC
NOP					*									------
ORA	*	*	*			*	*	*	*	*				N---Z-
PHA					*									------
PHP					*									------
PLA					*									N---Z-
PLP					*									NVDIZC
ROL		*	*	*				*	*					N---ZC
ROR		*	*	*				*	*					N---ZC
RTI					*									NVDIZC
RTS					*									------
SBC	*	*	*			*	*	*	*	*				NV--ZC
SEC					*									-----C
SED					*									--D---
SEI					*									---I--
STA		*	*			*	*	*	*	*				------
STX		*	*										*	------
STY		*	*					*						------
TAX					*									N---Z-
TAY					*									N---Z-
TSX					*									N---Z-
TXA					*									N---Z-
TXS					*									------
TYA					*									N---Z-

M. Zero-Page-indizierte Adressierung mit Indexregister Y

Beispiel: LDX $D0,Y
Befehlslänge: Zwei Byte
Taktzyklen: Vier

Indexregister ist Y. Ansonsten gilt das bereits über diese Adressierungsart mit Indexregister X Gesagte.
Die *Übersichtstabelle* gibt Auskunft, welche Adressierungsarten für welche Operationen bereitstehen. Sollte der Assembler einmal einen Befehl nicht annehmen wollen, wie LDA $D0,Y, so belehrt uns die Tabelle sogleich, daß es eine Zero-Page-indizierte Adressierung mit Y-Register nur für LDX und STX gibt. Ein paar Spalten weiter links sagt sie uns, daß die direkt indizierte Adressierung jedoch möglich ist, und wir mit LDA $00D0,Y zum Ziel kommen.

Beeinflussung der Statusflags

Zur Erstellung der Programmlogik ist es notwendig, zu wissen, welcher Befehl welches Statusflag beeinflußt. In der rechten Spalte unserer Übersichtstabelle findet sich diese Information. Auf die Darstellung von Bit vier und fünf des Statusregisters haben wir verzichtet, da Bit fünf keine Funktion hat und Bit vier, das Break-Bit, nur bei einem einzigen Befehl gesetzt wird, dem BRK-Befehl.
Dieses Flag wird im Betriebssystem unseres Rechners nur ein einziges Mal beim Interrupt abgefragt. Ist dieses Bit dort gesetzt, wird über den Sprungvektor $032C der Maschinenmonitor aufgerufen. Wollen wir den Sprung in den Monitor durch eine andere Routine ersetzen, so genügt eine Umstellung des Break-Vektors. Um das Break-Bit brauchen wir uns also nicht zu kümmern, wenn wir nicht selbst ein vollkommen neues Betriebssystem schreiben wollen.
Wir ersehen aus der Tabelle, daß die Statusflags bei Sprüngen, ob bedingten, absoluten oder Unterroutinen-Aufrufen, nicht beeinflußt werden. Ebenfalls keine Änderungen treten beim Schreiben von Registerinhalten in Speicherstellen auf, ob dieses nun mit Store-Befehlen, wie STA, STX, STY oder mit den Stapelbefehlen PHA und PHP geschieht.
Für den Registertransfer TSX, BRK, dem Sprung in den Break-Interrupt, und NOP, das gar nichts macht, gilt dasselbe.
Die spezifischen Set- und Clear-Befehle für die einzelnen Statusflags ändern nur das jeweilige Bit, auf das sie sich beziehen. PLP und RTI stellen das ehemals gerettete Statusflag wieder her. Die sonstigen Befehle beeinflussen das Vorzeichen-Bit N und das Zero-Flag Z. Bei Shift-, Rotate-Befehlen, Addition, Subtraktion und Vergleichen wird zusätzlich das Carry-Flag beeinflußt. Addition, Subtraktion und der Bit-Befehl wirken sich auch auf das Überlauf-Flag V aus.

VIII. EIN- UND AUSGABEROUTINEN

Über eine Standard-Sprungtabelle, das Kernel, kann jedes Ein- und Ausgabegerät angesprochen werden. Das Kernel sorgt dafür, daß Maschinenprogramme, die sonst keine systemspezifischen Adressen verwenden, auf jedem Acht-Bit-Commodore-Rechner lauffähig sind.

Was nutzt ein Rechner, der eine CPU besitzt, über ROM und RAM verfügt, jedoch keine Möglichkeiten bietet, Daten ein- oder auszugeben? Ein Rechner, der nicht die Möglichkeit bietet, mit seiner Umgebung zu kommunizieren, ist zu überhaupt nichts nütze. Wir benötigen daher noch diverse Ein- und Ausgabeoperationen. Für die Kommunikation mit der Außenwelt besitzt der Rechner verschiedene I/O-Bausteine, die über bestimmte Adressen angesprochen werden können.
Da für das Ansprechen solcher Bausteine ziemliche Kenntnisse über dieselben und über den Aufbau des bestimmten Rechners erforderlich

wären, haben die Systemprogrammierer bereits die erforderlichen Routinen programmiert.
Ein solches Betriebssystem muß vorhanden sein, sonst könnten wir gar kein Programm in den Rechner bringen. Nicht einmal ein einziger Tastendruck würde vom Rechner verarbeitet werden können.
Der Teil des Betriebssystem, der für Ein- und Ausgaben verantwortlich ist, heißt *BIOS*, was **B**asic **I**nput **O**utput **S**ystem bedeutet. Das Wort Basic hat nichts mit der Programmiersprache BASIC zu tun, sondern sagt aus, daß es hier um einfache Ein- und Ausgaberoutinen geht, die die Grundlage für die Kommunikation mit der Peripherie bilden, und auf die alle komplexeren Ein- und Ausgaberoutinen wie beispielsweise der PRINT-Befehl oder der INPUT-Befehl zurückgreifen.
Um auch bei unterschiedlich organisierten Rechnern dafür zu sorgen, daß diese in gleicher Weise zu handhaben sind, wird der Aufruf dieser Funktionen standardisiert. Beim VC20, C64, C16/116/Plus4 und dem C128 gibt es einen Speicherbereich im ROM, der an genau denselben Stellen eine Sprungliste, das Kernel, für diese wichtigen Systemaufrufe enthält. Um ein Zeichen aus dem Tastaturpuffer zu lesen, brauchen wir deshalb nicht zu wissen, wo dieser in unserem Rechner ist, und wie er verwaltet wird. Es existiert schließlich schon die Routine, die dies für uns erledigt. Wo diese Routine in unserem Rechner genau liegt, braucht uns ebenfalls nicht zu interessieren. Alles, was wir wissen müssen, ist, daß bei Ansprung der Adresse $FFE4 der Rechner in die gewünschte Routine verzweigt.

Eingaberoutinen

Zwei fundamentale Routinen besitzt unser Rechner, mit denen wir Daten von einem externen Gerät lesen können:

GETIN ($FFE4)

Mit dieser Routine können wir Daten von einem Eingabegerät einlesen. Die BASIC-Befehle GET und GET# bauen darauf auf. Wurde zuvor kein anderes Gerät als Eingabegerät definiert, so wird ein Zeichen aus dem Tastaturpuffer oder dem Funktionstastenpuffer an den Akku den Wert #$00 zugewiesen.

```
JSR $FFE4
```

Im Akku sollte sich nun der Wert #$00 befinden. Wir geben den Befehl jetzt erneut ein. Da eine Wiederholung vorliegt, brauchen wir hierzu nur die Return-Taste zu drücken. Gleich nach dem Druck dieser Taste, sollten wir blitzschnell noch eine andere Taste drücken. Bis das in BASIC verfaßte CPU-Programm in das Maschinenprogramm verzweigt, sollte der ASCII-Wert der gedrückten Taste bereits im Tastenpuffer abgelegt sein. Der Akkumulator zeigt im Anschluß diesen Wert. GETIN wartet nicht auf einen Tastendruck, wie dieses der BASIC-Befehl GETKEY tut. Da je nach erhaltenem Wert auch das Statusflag beeinflußt wird, können wir das Warten auf einen Tastendruck leicht selbst programmieren.

```
BRK
A301D JSR $FFE4
BEQ $301D
RTS
G
RUN
JSR $301D
```

BASIN ($FFCF)

Auch BASIN liest sowohl von der Tastatur, wie auch von sonstigen Eingabegeräten. Ist nicht die Tastatur das Eingabegerät, so besteht zwischen GETIN und BASIN kein Unterschied. Beim Lesen von der Tastatur ist der Unterschied nicht zu übersehen.

```
JSR $FFCF
```

Deutlich wird hier, daß BASIN die Grundlage des INPUT-Befehls bildet. Es erscheint ein blinkender Cursor und wir können soviel Zeichen eingeben, wie wir gerne möchten. Erst beim Druck der Return-Taste wird die BASIN-Routine beendet und wir bekommen das an erster Stelle eingegebene Zeichen in den Akku. Wie bekommen wir aber nun die restlichen Zeichen? Da das CPU-Programm hierauf keine Antwort zu geben vermag, so sei Ihnen dieses jetzt verraten. Bei jedem folgenden Aufruf von BASIN wird das nächste Zeichen in den Akku geladen, bis schließlich das Ende der Eingabe erreicht ist, was der Akku uns mit #$0D für das Returnzeichen anzeigt. Eingegebene Leerstellen am Ende werden ignoriert. Das CPU-Programm stört diese Eigenheit von BASIN. Deshalb verfassen wir ein kleines Maschinenprogramm, mit dem wir Eingaben nach Wahl im Hauptspeicher unseres Rechners positionieren können. Wie dieses geht, sollten Sie nun schon wissen. Hier ist das disassemblierte Monitorlisting.

```
. 3023  84 d0      sty  $d0
. 3025  86 d1      stx  $d1
. 3027  a0 00      ldy  #$00
. 3029  20 cf ff   jsr  $ffcf
. 302c  91 d0      sta  ($d0),y
. 302e  e6 d0      inc  $d0
. 3030  d0 02      bne  $3034
. 3032  e6 d1      inc  $d1
. 3034  c9 0d      cmp  #$0d
. 3036  d0 f1      bne  $3029
. 3038  a4 d0      ldy  $d0
. 303a  a6 d1      ldx  $d1
. 303c  60         rts
```

Um das Programm verschiebbar zu gestalten, benutzten wir die indirekte Adressierung. Um einen größeren Bereich als 256 Byte ansprechen zu können, und nicht am Ende extra den Wert des Y-Registers zum Inhalt der Speicherstelle $D0 addieren zu müssen, verzichteten wir auf eine Indizierung, indem wir das Y-Register kurzerhand auf Null setzten. Schnell genug ist das Programm wohl schon auch so.
Vor dem ersten Aufruf dieses Programmes ist dem X-Register das High-Byte der Speicheradresse und dem Y-Register das Low-Byte mitzuteilen. Die Eingabe wird beim Aufruf ab dieser Adresse gespeichert. Sowohl die Speicherzellen $D0 und $D1, als auch die Indexregister weisen hinterher auf die den abgespeicherten Daten folgende Speicherstelle. Durch mehrmaligen Aufruf dieses Maschinenprogrammes können so kontinuierlich weitere Abspeicherungen vorgenommen werden. Sollten bei zwischendurch erfolgenden Operationen die Indexregisterinhalte verlorengehen, so braucht die Eingabe das nächste Mal anstatt mit JSR $3023 nur mit JSR $3027 angesprungen zu werden. Wir probieren in unserem CPU-Programm ein paar Eingaben.

```
LDX #$31
LDY #$00
JSR  $3023
```

Durch Druck der Returntaste können wir weitere Eingaben vornehmen.

```
BRK
M3100
G
```

Im Monitor können wir uns vom Funktionieren der Routine überzeugen.

Die Ausgaberoutine BSOUT ($FFD2)

BSOUT bildet die Grundlage für alle Ausgaben. Der im Akku befindli-

che Wert wird an das Ausgabegerät gesandt.
Sofern nichts anderes definiert wurde, ist dieses der Bildschirm.

```
LDA #$41
JSR  $FFD2
```

Auf dem Bildschirm sollte der Buchstabe A erscheinen. Bei einem längeren auszugebenden Text ist jedesmal ein Zeichen in den Akku zu laden und es dann mit BSOUT auszugeben, ist die schnellste, aber natürlich nicht die richtige Methode. Genauso, wie bei der Eingabe, verfassen wir daher eine Routine, die Daten ausgibt, welche in irgendeinem Speicherbereich liegen.

```
. 303d  84 d2      sty  $d2
. 303f  86 d3      stx  $d3
. 3041  a0 00      ldy  #$00
. 3043  b1 d2      lda  ($d2),y
. 3045  20 d2 ff   jsr  $ffd2
. 3048  e6 d2      inc  $d2
. 304a  d0 02      bne  $304e
. 304c  e6 d3      inc  $d3
. 304e  c9 0d      cmp  #$0d
. 3050  d0 f1      bne  $3043
. 3052  a6 d3      ldx  $d3
. 3054  a4 d2      ldy  $d2
. 3056  60         rts
```

Oft erfolgen Ein- und Ausgaben abwechselnd. Damit keine gegenseitige Verstellung der Zeiger auftritt. Haben wir die Adressen $d2 und $d3 als Ausgabezeiger verwendet. Soll beim erweiterten C16 oder Plus4 auch der gebankte RAM-Speicher ab Adresse $8000 angesprochen werden, so wäre die Routine etwas zu verändern. Dieses Thema werden wir aber erst später behandeln. Nun wollen wir die vorher eingegebenen Daten auf den Bildschirm ausgeben.

```
LDX #$31
LDY #$00
JSR  $303C
JSR  $303C
JSR  $303C
```

Ansteuerung externer Geräte

OPEN, CLOSE, und CMD gibt es auch für Maschinensprache. Zur Parameterübergabe an den OPEN-Befehl bedarf es aber erst einiger vorbereitetender Routinen.

SETLFS ($FFBA)

Mit SETLFS werden die logische Dateinummer, die Geräte- und die Sekundäradresse erfaßt. Wir laden die Dateinummer in den Akku, die Geräteadresse in das X-Register und die Sekundäradresse in das Y-Register.

Beispiel: OPEN1,4,7

```
LDA #$01
LDX #$04
LDY #$07
JSR  $FFBA
```

Soll keine Sekundäradresse angegeben werden, so ist Y mit #$FF zu laden.

SETNAM ($FFBD)

Um bei Kassetten- oder Diskettenoperation auch einen Dateinamen angeben zu können, existiert die SETNAM-Routine. Hierbei wird dem Akku die Länge des Dateinamens, dem X-Register das Low-Byte der Adresse, wo der Dateiname sich befindet, und dem Y-Register das Adreß-High-Byte mitgeteilt. Beim Drucker ist kein Dateiname erforderlich. In diesem Falle geben wir die Länge des Dateinamens mit Null an. Um die Adresse brauchen wir uns dann nicht mehr zu kümmern.

```
LDA #$00
JSR  $FFBD
```

OPEN ($FFC0)

Nachdem die Parameter durch SETLFS und SETNAM festgelegt sind, können diese durch OPEN in der File-Tabelle des Rechners vermerkt werden.

```
JSR  $FFC0
```

Bei Floppy oder Datasette werden diese Geräte zusätzlich angesteuert, um entweder den entsprechenden File zu suchen oder anzulegen.

CHKOUT ($FFC9)

Dieser Befehl entspricht dem CMD-Befehl, der uns bereits vom BASIC her bekannt ist. Nachdem ein entsprechender File mit OPEN eröffnet wurde, können wir das Ausgabegerät umdefinieren. Hierzu ist das X-Register mit der Kanalnummer zu laden und CHKOUT aufzurufen.

```
LDX #$01
JSR  $FFC9
```

Daten, die jetzt mit BSOUT ausgegeben werden, erscheinen nun nicht mehr auf dem Bildschirm. Spätestens beim ersten Zeichen nach einem Return mit Code #$0D sollte der Drucker mit dem Druck beginnen.

```
LDA #$41       LDA #$42
JSR  $FFD2     JSR  $FFD2
LDA #$0D       LDA #$0D
JSR  $FFD2     JSR  $FFD2
```

Der Buchstabe A wird jetzt in jedem Fall auf dem Drucker ausgegeben. Wenn der Buchstabe B noch nicht erscheint, so tut er dies bestimmt, wenn der Druckerkanal wieder geschlossen wird.

CLRCHN ($FFCC)

Um wieder die Tastatur als Eingabegerät und den Bildschirm als Ausgabegerät zu definieren, ist keine Parameterübergabe, sondern nur der Aufruf von CLRCHN nötig.

```
JSR  $FFCC
LDA #$41
JSR  $FFD2
```

Das Zeichen A erscheint wieder auf dem Bildschirm. Mit CHKOUT kann jederzeit wieder auf den Drucker umgeschaltet werden.

CLOSE ($FFC3)

Wurde ein Kanal mit CLRCHN geschlossen, so können wir, sofern wir ihn nicht mehr benötigen sollten, ganz aus unserer File-Tabelle beseitigen. Hierzu ist der Akkumulator mit der Kanalnummer zu laden.

```
LDA #$01
JSR  $FFC3
```

Mit CHKOUT auf den Drucker umschalten zu wollen, ist jetzt zwecklos. Denn der Rechner kennt die Geräteparameter nicht mehr. Erst nach erneutem Anlegen mit SETLFS, SETNAM und OPEN bekommen wir unseren Drucker wieder in den Griff. Der CLOSE-Befehl ist besonders wichtig bei Kassetten- und Diskoperationen, damit Floppy- und Datasettendatei ordentlich geschlossen werden.

CLALL ($FFE7)

Ziemlich radikal ist diese Routine. Wie CLRCHN schließt sie den jeweiligen Gerätekanal. Zusätzlich beseitigt sie aber alle Einträge aus der Filetabelle. Wenn ein Gerät noch ordentlich mit CLOSE abgeschlossen werden sollte, gibt es Probleme, da die Geräteparameter und der Dateiname nicht mehr greifbar sind.

CHKIN ($FFC6)

Wie CHKOUT einen Ausgabekanal öffnet, so tut dieses CHKIN mit einem Eingabekanal. Mit GETIN oder BASIN können die Daten dann vom entsprechenden Gerät eingelesen werden. Das Öffnen eines Eingabekanals hat keinen Einfluß auf den Ausgabekanal und umgekehrt. Daten können so von ei-

nem externen Gerät eingelesen und mit BSOUT gleich wieder auf das Ausgabegerät ausgegeben werden. Die Parameterübergabe erfolgt bei CHKIN ebenfalls über das X-Register.

READST ($FFB7)

Von BASIC her ist uns die Statusvariable bereits bekannt. Wenn READST aufgerufen wird, bekommen wir die Statusvariable in den Akku. Daraus können wir ersehen, ob das Ende einer Datei erreicht ist, oder sonstige Fehler vorliegen.

Ein- und Ausgabe-Experimente

Wenn unsere beiden Maschinenprogramme für Ein- und Ausgabe noch im Speicher vorliegen,so können wir ein wenig experimentieren.

Datei für Drucker anlegen

```
LDA #$ 04
TAX
LDY #$07
JSR $FFBA
LDA #$00
JSR $FFBD
JSR $FFC0
```

Daten erfassen

```
LDA #$31
LDY #$00
JSR $3023
Eingabe nach Wahl
JSR $3023
Eingabe nach Wahl
JSR $3023
Eingabe nach Wahl
```

Daten auf Drucker ausgeben

```
LDX #$04
JSR $FFC9
LDX #$31
LDY #$00
JSR $303D
JSR $303D
JSR $303D
JSR $FFCC
```

Dateiname erfassen

```
LDX #$38
LDY #$00
JSR $3023
TESTDATEI,S,W
```

Diskettendatei eröffnen

```
LDA #$08
TAX
TAY
JSR $FFBA
LDA #$0D
LDX #$00
LDY #$38
JSR $FFBD
JSR $FFC0
```

Auf Diskette speichern

```
LDX #$ 08
JSR FFC9
LDX #$31
LDY #$00
JSR $303D
JSR $303D
JSR $303D
JSR $FFCC
LDA #$08
JSR $FFC3
```

Da die Daten nun gespeichert sind, können wir in den Monitor gehen und sie löschen.

```
BRK
F3100,3FFF,0
M3100
G
```

Wir führen diese Operation durch, damit niemand sagen kann, wir würden jetzt die Daten aus dem Hauptspeicher lesen.

Dateiname eingeben

```
LDX #$38
LDY #$00
JSR $3023
TESTDATE,S,R
```

Diskettendatei eröffnen

```
LDA #$08
TAX
TAY
JSR $FFBA
LDA #$0D
LDX #$00
LDY #$38
JSR $FFBD
JSR $FFC0
```

Daten einlesen und ausgeben

```
LDX #$08
JSR $FFC6
LDX #$31
LDY #$00
STX $00D3
STY $00D2
JSR $3023
JSR $3041
JSR $3023
JSR $3041
JSR $3023
JSR $3041
JSR $FFCC
LDA #$08
JSR $FFC3
JSR $FFE7
```

Ein Blick in den Monitor belehrt uns zusätzlich, daß unsere Daten nun auch wieder im Hauptspeicher vorhanden sind.

```
BRK
M3100
G
```

Damit sind wir schon fast am Ende unserer Einführung angelangt. Das Speichern und Laden von Maschinenprogrammen und das Wandeln in DATA-Zeilen sollen noch kurz abgehandelt werden.

Laden und Speichern von Maschinenprogrammen

Das Speichern geschieht am besten vom Monitor aus. Wir speichern den Bereich ab, in dem das Maschinenprogramm steht. Zuerst ist der Buchstabe S für Save einzugeben. Diesem folgt zwischen Anführungszeichen der Filename. Durch Komma getrennt geben wir die Gerätenummer ein. Nach einem weiteren Komma folgt die Anfangsadresse und zuallerletzt, ebenfalls durch Komma getrennt, die Endadresse. Die Endadresse muß um eine Speicherstelle höher angegeben werden. Der folgende Befehl würde so zum Beispiel den Bereich von $3000 bis $30FF auf Diskette sichern.

```
S"TESTFILE",8,3000,3100
```

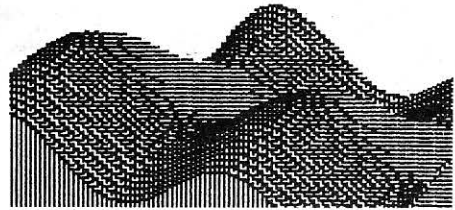
Beim Laden erübrigen sich die Adreßangaben.

```
L"TESTFILE",8
```

Bei der Kassette ist anstelle einer Acht die Eins als Gerätenummer zu verwenden.

Wandeln in DATA-Zeilen

Für kurze Maschinenroutinen, die in ein BASIC-Programm eingebunden werden sollen, empfiehlt sich die Umwandlung in DATA-Zeilen. Die DATA werden vom BASIC-Programm wieder in die ursprünglichen Speicherstellen gePOKEd. Ein kurzes Programm *Datawandler* besorgt die Umwandlung. Nach dem Start ist die Zeilennummer einzugeben, ab der die DATA abgelegt werden sollen. Danach will das Programm Anfang und Ende des umzuwandelnden Speicherbereiches in hexadezimaler Form wissen. Die erzeugten Zeilen werden nur auf dem Bildschirm ausgegeben. Erst wenn Sie mit dem Cursor auf die zu sehenden Zeilen fahren und sie mit Return aufnehmen, stehen diese im Programm zur Verfügung. Mit dem DELETE-Befehl brauchen Sie nur noch die Programmzeilen des Datawandlers zu löschen. *AM*□



```
10 rem cpu-trainer=============c16 <de>
20 rem (p) commodore welt team <ho>
30 rem ========================== <ng>
40 rem (c) by alfons mittelmeyer <cg>
50 rem <pd>
60 rem <ah>
70 rem basic v3.5 <nl>
80 rem c16/116/plus4 <ki>
90 rem ========================== <jg>
100 poke55,0:poke56,48:clr <bm>
110 fori=1630to1746 <jp>
120 reada:pokei,a:next <ag>
130 data 173,214,006,072,173,211 <fn>
140 data 006,174,212,006,172,213 <li>
150 data 006,040,076,050,003,141 <hk>
160 data 211,006,142,212,006,140 <db>
170 data 213,006,008,104,141,214 <if>
180 data 006,088,216,096,173,211 <om>
190 data 006,174,215,006,157,000 <jp>
200 data 001,206,215,006,096,238 <el>
210 data 215,006,174,215,006,173 <ef>
220 data 214,006,072,040,189,000 <fd>
230 data 001,141,211,006,008,104 <no>
240 data 141,214,006,088,216,096 <oi>
250 data 173,214,006,024,144,215 <dm>
260 data 238,215,006,174,215,006 <kd>
270 data 189,000,001,141,214,006 <nn>
280 data 096,173,214,006,072,040 <jd>
290 data 173,215,006,141,212,006 <pb>
300 data 008,104,141,214,006,088 <ni>
310 data 024,096,173,212,006,141 <ik>
320 data 215,006,096 <no>
330 sys1647:poke1751,127 <nj>
340 rem -------------------------- <ip>
350 rem  variablen <oo>
360 rem -------------------------- <gg>
370 b$=chr$(32):b4$=b$+b$+b$+b$ <cg>
380 a$(0)="aiiiiiiiib" <cd>
390 a$(1)="j00000000j" <ff>
400 a$(2)="diiiiieiif" <kh>
410 a$(3)="000000j00j" <dl>
420 a$(4)="00000agiif" <eh>
430 a$(5)="00000j000j" <en>
440 a$(6)="00000diiic" <bc>
450 data 032,176,174,189,173 <hf>
460 data 178,179,177,171,192,221 <fl>
470 g$="":fori=0to10:reada:g$=g$+c
hr$(a):next <il>
480 forj=0to6:fori=1to10 <oe>
490 mid$(a$(j),i,1)=mid$(g$,((asc(
mid$(a$(j),i,1))and15))+1,1):next:
nextj <pp>
500 b$(0)="binaer":b$(1)="hex" <cp>
510 b$(2)="dezimal" <gd>
520 c$="akku"+b4$+"x-register y-re
gister" <le>
530 d$="nv--dizc"+b4$+"stapel" <fh>
540 rem -------------------------- <eg>
550 rem  bildschirmaufbau <mp>
560 rem -------------------------- <eh>
570 char,0,0,"":scnclr <jo>
580 gosub910:y=3:gosub900 <ll>
590 char,10,1,c$:x=7:y=2:gosub890 <kk>
600 x=18:gosub890:x=29:gosub890 <mn>
610 y=12:gosub900 <ea>
620 char,8,10,d$:x=7:y=11:gosub890 <of>
630 x=18:gosub890:char,0,18,"":gos
ub910 <op>
640 x$="":gosub950 <ml>
650 rem -------------------------- <fi>
660 rem  eingabe <mc>
670 rem -------------------------- <mk>
680 ifx$="brk"then570 <ig>
690 char,0,19,chr$(27)+"q? "+chr$(
164) <ji>
700 y$=" ":x=1 <id>
710 sys60381:a=peek(2034):if(a<32a
nda<>13anda<>20)ora>95then710 <le>
720 ifa=13then800 <fj>
730 ifa<>20theny$=y$+chr$(a):x=x+1
:goto760 <hk>
740 ifx=1then700 <an>
750 x=x-1:y$=left$(y$,x) <ja>
760 poke2034,157:sys56393:poke2034
,a:sys56393:poke203,0:poke2034,164
:sys56393:goto710 <ga>
770 rem -------------------------- <ob>
780 rem  eingabe bearbeiten <mg>
790 rem -------------------------- <io>
800 ify$<>" "thenx$=right$(y$,x-1) <kh>
810 poke2034,13:sys56393 <oi>
820 ifx$="brk"thenscnclr <hp>
830 poke2034,27:sys56393:poke2034,
84:sys56393 <pg>
840 poke2034,147:sys56393 <jc>
850 gosub1180:goto680 <gj>
860 rem -------------------------- <ie>
870 rem  routinensammlung <pe>
880 rem -------------------------- <ad>
890 fori=0to6:char,x,y+i,a$(i):nex
t:return <pc>
900 fori=0to2:char,0,y+i+i,b$(i):n
ext:return <lb>
910 poke2034,61:fori=1to40:sys5639
3:next:return <ch>
920 data 1747,7,2,1748,18,2 <dl>
930 data 1749,29,2,1750,7,11 <hk>
940 data 1751,18,11 <gh>
950 ifx$="brk"thenreturn <fj>
960 restore920:forz=1to5:reada <bd>
970 readx:ready:a=peek(a) <nf>
980 char,x+1,y+1,"":s=a:n=128 <mb>
990 fori=1to8:a$="0" <ma>
1000 ifs>=nthens=s-n:a$="1" <nd>
1010 char,x+i,y+1,a$:n=n/2:next <bl>
1020 char,x+7,y+3,right$(hex$(a),2
```

```
)                                       <ep>
1030 char,x+6,y+5,right$(" "+str$(
a),3)                                   <dl>
1040 nextz:return                       <mb>
1050 x=-1:fori=1tom:reada$:reada:i
fa$=y$theni=50                          <em>
1060 next:return                        <en>
1070 char,0,19,"fehler":poke239,0:
wait239,1:return                        <ja>
1080 rem ------------------------       <fh>
1090 rem  ein-byte-befehle              <bm>
1100 rem ------------------------       <om>
1110 data asl,10,brk,0,clc,24           <ab>
1120 data cld,216,cli,88,clv,184        <kg>
1130 data dex,202,dey,136,inx,232       <lf>
1140 data iny,200,lsr,74,nop,234        <nn>
1150 data rol,42,ror,106,sec,56         <if>
1160 data sed,248,sei,120,tax,170       <pg>
1170 data tay,168,txa,138,tya,152       <cg>
1180 iflen(x$)>4then1340                <pg>
1190 y$=x$:restore1110:m=21:gosub1
050                                     <gc>
1200 ifi=51thenpoke1644,a:poke1645
,234:poke1646,234:sys1630:goto950       <mh>
1210 rem ------------------------       <pj>
1220 rem  sonderbehandlung              <mg>
1230 rem ------------------------       <ib>
1240 m=6:gosub1050:ifi<>51then1070      <dh>
1250 sysa:goto950                       <ei>
1260 data pha,1664,pla,1677,php,17
02,plp,1708,tsx,1721,txs,1740           <kb>
1270 rem ------------------------       <le>
1280 rem  zwei-byte-befehle             <lk>
1290 rem ------------------------       <ii>
1300 data adc,105,and,41,cmp,201        <go>
1310 data cpx,224,cpy,192,eor,73        <lp>
1320 data lda,169,ldx,162,ldy,160       <kc>
1330 data ora,9,sbc,233                 <gf>
1340 ifleft$(right$(x$,4),2)<>"#$"
then1510                                <on>
1350 restore1300:y$=left$(x$,3)         <ig>
1360 m=11:gosub1050:ifi<>51then107
0                                       <mc>
1370 x=dec(right$(x$,2))                <jg>
1380 poke1644,a:poke1646,234            <bk>
1390 poke1645,x:sys1630:goto950         <ah>
1400 rem ------------------------       <ea>
1410 rem  drei-byte-befehle             <gp>
1420 rem ------------------------       <jc>
1430 data adc,109,and,45,asl,14         <eh>
1440 data bit,44,cmp,205,dec,206        <nc>
1450 data eor,77,inc,238,jsr,32         <fh>
1460 data lda,173,ldx,174,ldy,172       <he>
1470 data lsr,78,ora,13,rol,46          <in>
1480 data ror,110,sbc,237,sta,141       <hg>
1490 data stx,142,sty,140,cpx,224       <ll>
1500 data cpy,192                       <em>
1510 ifleft$(right$(x$,5),1)<>"$"t
hen1070                                 <ba>
1520 restore1430:y$=left$(x$,3)         <if>
1530 m=22:gosub1050:ifi<>51then107
0                                       <cg>
1540 y$=right$(x$,4)                    <oh>
1550 poke1646,dec(left$(y$,2))          <fn>
1560 poke1645,dec(right$(y$,2))         <in>
1570 poke1644,a                         <gn>
1580 sys1630:goto950                    <gi>
1590 rem ========================       <mi>
1600 rem 12277 bytes memory             <ph>
1610 rem 04414 bytes program            <ch>
1620 rem 00105 bytes variables          <cg>
1630 rem 00080 bytes arrays             <fb>
1640 rem 00451 bytes strings            <ih>
1650 rem 03087 bytes free (0)           <mh>
1660 rem 04096 bytes reserviert         <nn>
1670 rem ========================       <jd>
```

DATAWANDLER

```
1 rem datawandler==============c16      <pa>
2 rem (p) commodore welt team           <lm>
3 rem ===========================       <cd>
4 rem (c) by alfons mittelmeyer         <ce>
5 rem                                   <jj>
6 rem                                   <jl>
7 rem basic v3.5                        <ni>
8 rem c16/116/plus4                     <jk>
9 rem ===========================       <nd>
10 input"zeilennummer";zn               <cg>
11 input "anfang,ende";x$,y$            <pf>
12 x=dec(x$):y=dec(y$)                  <mf>
13 x$=str$(x):y$=str$(y)                <hc>
14 x$=right$(x$,len(x$)-1)              <ca>
15 y$=right$(y$,len(y$)-1)              <ng>
16 gosub29:print" fori="x$"to"y$:z
n=zn+10                                 <jc>
17 gosub29:print" reada:pokei,a:ne
xt"                                     <ac>
18 zn=zn+10                             <bf>
19 l=int((y-x+1)/6)+1:k=(y-x+1)-(l
-1)*6                                   <dm>
20 ifk=0thenl=l-1:k=6                   <hh>
21 forj=1tol:n=6:ifj=lthenn=k           <ip>
22 gosub29:print" data ";               <lh>
23 for i=0 to n-1:a=peek(x+i)           <kj>
24 a$=str$(a):a$=right$(a$,len(a$)
-1)                                     <ef>
25 a$=right$("00"+a$,3)                 <cc>
26 print a$",";:next:printchr$(20)      <li>
27 x=x+6:zn=zn+10:next                  <ec>
28 end                                  <li>
29 printright$(str$(zn),len(str$(z
n))-1);                                 <mi>
30 return                               <mk>
31 rem ===========================      <kh>
32 rem  p r o g r a m m e n d e         <mb>
33 rem ===========================      <md>
```

SEQ-FILES

Lesen von ASCII-Texten

Nicht immer ist eine Textverarbeitung zur Hand, mit der auf Diskette oder Kassette vorliegende Text-Files gelesen werden können. Das Programm SEQ-FILES sorgt für Abhilfe.

Nicht nur Programme können auf Diskette gespeichert werden, sondern auch Nachrichten in Form von sequentiellen Files. Oft besteht die Schwierigkeit, daß derjenige, der eine solche Nachricht erhält, vielleicht nicht die Textverarbeitung benützt, die der Verfasser des Textes verwandt hatte. Die Folge kann sein, daß er die Nachricht nicht lesen kann.

Ein Standard für Textdateien ist die sequentielle ASCII-Datei. Bereits durch Öffnen einer sequentiellen Datei und entsprechende PRINT#-Befehle kann Text abgespeichert werden. Mit einigen Textverarbeitungen wie zum Beispiel Script-Plus läßt sich dieser Text einlesen. Die integrierte Software des Plus4 braucht aber nicht die ASCII-, sondern die Bildschirm-Codes. Manch andere Textverarbeitung mag wohl auch mit einem speziellen Datenformat arbeiten. Eine sequentielle ASCII-Text-Datei sollte aber von jedermann zu lesen sein.

Das Programm SEQ-Files funktioniert sowohl für Kassette als auch für Diskette. Bei der Kassette kann die Frage nach dem File-Namen mit Return beantwortet werden, während die Diskette den File-Namen braucht. Der Text wird in einer Schleife eingelesen und auf den Bildschirm geschrieben. Eine Zeitschleife bremst die Geschwindigkeit, so daß mitgelesen werden kann. Durch Druck der Leertaste kann das Programm jederzeit angehalten und wieder gestartet werden, wenn ein längeres Betrachten erwünscht ist. Die Kassettenbenutzer müssen in Kauf nehmen, daß der Bildschirm sich nach der Ausgabe von jeweils 192 Zeichen abschaltet, damit die Datasette den nächsten Block nachladen kann. Von Kassette lädt der Rechner nämlich einen Block von 192 Zeichen in den Kassettenpuffer. Der Bildschirm wird nach einem solchen Ladevorgang wieder eingeschaltet. Die folgenden GET#- oder INPUT#-Befehle greifen auf den Kassettenpuffer zu, bis alle Zeichen darin gelesen sind. Dann erfolgt ein neuer Ladevorgang. Ist der Text restlos gelesen, was sich an der Statusvariablen ST erkennen läßt, wird das Programm beendet. □

SEQFILES READ

```
10 rem seqfiles read ==========cbm  <nf>
20 rem (p) commodore welt team      <ho>
30 rem ==========================   <ng>
40 rem (c) by alfons mittelmeyer    <cg>
50 rem                              <pd>
60 rem                              <ah>
70 rem version 2.0  40/80z ascii    <of>
80 rem 20/64/16/116/p4/128/cbm      <jb>
90 rem ==========================   <jg>
100 printchr$(147)chr$(142)"commod
ore welt team"                      <em>
110 print:print"einlesen von"       <ol>
120 print"sequentiellen-files":pri
nt                                  <cj>
130 print"1 = lesen von kassette 0
1"                                  <ho>
140 print"2 = lesen von diskette 0
8"                                  <an>
150 getx$:ifx$=""then150            <eg>
160 onval(x$)goto200,210            <hn>
170 goto150                         <fe>
180 rem ******* filename eingabe *  <id>
190 print:input"filename: ";a$:ret
urn                                 <me>
200 gosub190:open8,1,0,a$:goto220   <np>
210 gosub190:open8,8,8,a$+",s,r"    <cl>
220 printchr$(147)chr$(14);         <kf>
230 get#8,x$:printx$;               <de>
240 getx$:ifx$=""then270            <mf>
250 getx$:ifx$=""then250            <el>
260 rem ***** time monitor write *  <hm>
270 fori=1to30:next                 <ka>
280 ifst=0then230                   <ei>
290 printchr$(142);:close8          <mp>
300 rem seqfiles read =========cbm  <nc>
310 rem 001014 bytes program        <ld>
320 rem 000021 bytes variables      <oc>
330 rem 000020 bytes strings        <cb>
340 rem ========================    <ac>
```

MALPROGRAMM

Grafische Benutzeroberfläche mit Joystickzeiger

Das Malprogramm Paint System, das im HIRES-Modus arbeitet, erleichtert das Arbeiten durch eine GEM-ähnliche Benutzeroberfläche. Paint System bietet zahlreiche wichtige Funktionen zum Erstellen von Bildern und Zeichnungen.

Mit dem Joystick in Port eins (Port null am Plus4) wird die Funktion angewählt, die in einer Menüleiste angezeigt wird. Die Anwahl geschieht durch Drücken des Feuerknopfes. Zur Kontrolle erscheint die gewählte Funktion in einem Kontrollfeld, das durch M: gekennzeichnet ist. Der Zeiger erscheint im Arbeitsfeld. Am unteren Rand werden X- und Y-Position des Zeigers angezeigt. Durch einfaches Drücken der ESC-Taste kann die gewählte Funktion wieder verlassen werden, wonach der Pfeilzeiger wiederum in der Menüleiste zur Funktionswahl bereitsteht.

ERKLÄRUNGEN DER FUNKTIONEN (von oben links):

Nr. 1: **PENCIL (1. oben links)**
Dies ist ein ganz normaler Stift. Nach der Aktivierung durch nochmaliges Drücken auf den Feuerknopf kann wie mit einem Stift gezeichnet werden. Durch nochmaliges Drücken ist die Funktion wieder ausgeschaltet.

Nr. 2: **DEL PENCIL (1. unten links)**
Wie Nr. 1. Nur, daß dieser Stift löscht.

Nr. 3: **BOX (2. oben, links)**
Rechteck. Es werden zwei Punkte zur Zeichnung eines Rechteckes benötigt. Der erste gibt die rechte obere Ecke an, der zweite die linke untere Ecke. Beide werden durch Drücken des Feuerknopfes angegeben.

Nr. 4: **BLOCK (2. unten links)**
Wie Nr. 3. Nur wird das Rechteck ausgefüllt.

Nr. 5: **DEL BLOCK (3. oben links)**
Wieder werden rechte obere und linke untere Ecke angegeben. Dann wird dieser Teil gelöscht.

Nr. 6: **CIRCLE (3. unten links)**
Zuerst muß der Mittelpunkt angegeben werden, dann durch einen erneuten Druck auf den Feuerknopf der Radius.

Nr. 7: **DISC (4. oben links)**
Wie Nr. 6. Nur wird der Kreis gleich ausgefüllt.

Nr. 8: **FILL (4. unten links)**
Durch Drücken des Feuerknopfes wird von der momentanen Position an ausgefüllt.

Nr. 9: **SPRAY (5. oben links)**
Nach der Aktivierung dieser Funktion stehen vier verschiedene SPRAY-Stärken zur Verfügung, die durch Drücken der Tasten eins bis vier angewählt und aktiviert werden.

Nr. 10: **RAYS (5. unten links)**
Es wird durch Drücken des Feuerknopfes ein Mittelpunkt festgelegt. Von nun an wird nach jedem weiteren Drücken an beliebiger Stelle vom Mittelpunkt bis zur momentanen Position eine Linie gezogen.

Nr. 11: **LINE (6. oben links)**
Anfangspunkt und Endpunkt müssen angegeben werden. Dann wird eine Linie gezogen.

Nr. 12: **LINES (6. unten links)**
Ein Anfangspunkt wird festgelegt. Von nun an wird nach jedem Druck auf den Feuerknopf vom letzten bis zum aktuellen eine Linie gezogen.

Nr. 13: **COPY (7. oben links)**
Kopieren eines Bildschirmausschnittes. (Parameter ähnlich dem Rechteck).
Obere linke Ecke des Ausschnittes bestimmen, dann untere rechte Ecke festlegen.
Jetzt mit dem Pfeil an eine beliebige Stelle gehen und den Feuerknopf drücken (obere linke Ecke).
Hier wird der Bildschirmausschnitt hinkopiert.

Nr. 14: **LOAD (7. unten links)**
Einladen einer Grafik.

Nr. 15: **SAVE (8. oben links)**
Abspeichern einer Grafik.

Nr. 16: **DIRECTORY (8. unten links)**
Zeigen des Disketteninhalts.

Nr. 17: **SEND DISK COMMAND (9. oben links)**
Senden eines Diskettenbefehls. Command wie im OPEN-Befehl.

Nr. 18: **CLEAR (9. unten links)**
Löschen einer Grafik.

Nr. 19: **RESET (10. oben links)**
Nach einer Sicherheitsabfrage wird ein RESET durchgeführt.

Nr. 20: **TEXT (10. unten links)**
Nach Anwählen dieses Punktes und nochmaligem Drücken erscheint ein Cursor auf dem Grafikbildschirm. Mit ihm kann die Grafik ideal beschriftet werden. Grafikzeichen und reverse Schrift sind ebenfalls möglich. Mit den Cursor-Tasten kann der Cursor gesteuert werden. Mit der DEL-Taste wird das links stehende Zeichen gelöscht. Nach Drücken der HOME-Taste wird der Cursor an die linke obere Ecke gesetzt. Nach Drücken der RETURN-Taste befindet man sich wieder im normalen Zeichenmodus.

VERWENDUNG DER BILDER IN EIGENEN PROGRAMMEN

Möchte man ein Bild in eigenen Programmen verwenden, geht man am besten wie folgt vor:
Das Bild im Monitor einladen:
L"NAME DES BILDES",8
Den Monitor mit X verlassen und folgende Befehlsfolge eingeben:
GRAPHIC 1:GRAPHIC 0:BOX 1,0,178,319,199,,1:BOX 1,0,0,319,20,,1
Nun speichert man das Bild erneut im Monitor ab:
S"NAME DES BILDES",8,1800,4000
Das Bild ist jetzt fertig zum Programm-Gebrauch.

Sascha Haberlandt □

```
10 rem paint system=============p4 <fe>
20 rem (p) commodore welt team <ho>
30 rem ============================ <ng>
40 rem (c) by sascha haberlandt <bc>
50 rem <pd>
60 rem <ah>
70 rem basic v3.5 <nl>
80 rem plus4 (c16/116 + 64 kb) <fd>
90 rem ============================ <jg>
100 graphic1,1:clr:gosub 3970 <ko>
110 rem ************************** <lj>
120 rem * rem zeilen mit abtippen* <jj>
130 rem ************************** <na>
140 color0,2,5:color4,2,5:color1,1 <jn>
150 vol8:poke65286,peek(65286)and2
39:gosub3870 <df>
160 gosub3900:printcl$:draw1,0,0to
319,0:draw1,0,20to319,20:box1,0,17
9,319,199 <ea>
170 draw1,8,195to312,195 <ca>
180 fori=0to90step10:box1,i,0,i+10
,10 <fk>
190 box1,i,10,i+10,20:nexti:box1,1
00,0,120,10 <pd>
200 draw1,3,2to1,1to5,4to3,7to7,7t
o7,5 <dj>
210 draw1,1,12to2,13:draw1,4,15to5
,16 <jf>
220 draw1,7,18to9,20 <aj>
230 box1,12,3,18,7 <pk>
240 box1,12,13,18,17,,1 <ep>
250 draw1,22,3to24,3:draw1,26,3to2
7,3:draw1,27,3to27,5:draw1,27,7to2
7,8 <ck>
260 draw1,26,8to25,8:draw1,23,8to2
2,8:draw1,22,6to22,5 <ck>
270 circle1,25,15,2 <cf>
280 circle1,35,5,2:paint1,35,5 <nc>
290 box1,34,13,37,17,28,1 <ff>
300 draw1,44,4,46,3,46,5,44,5,47,6
,45,8,43,7,44,5 <jn>
310 draw1,45,15to42,12:draw1,45,15
to42,18:draw1,45,15to48,18:draw1,4
5,15to48,12 <kl>
320 draw1,42,15to48,15 <ic>
330 draw1,57,3to53,7 <hg>
340 draw1,52,12to52,18to56,12to58,
18 <gf>
350 box1,62,2,65,5:box1,64,5,67,8 <bi>
360 draw1,63,12to63,18to68,18 <bp>
370 draw1,78,2to74,2to73,3to74,4to
76,4to77,5to76,8to72,8 <fk>
380 draw1,73,12to73,18to76,18to78,
16to78,14to76,12to73,12 <jc>
390 draw1,84,2to82,2to82,5to84,5to
84,8to82,8:draw1,86,2to86,8to88,6t
o88,4to86,2 <pf>
400 draw1,87,13to86,12to84,12to83,
14to83,16to84,18to86,18to87,17 <el>
410 box1,92,2,98,8:box1,94,4,96,6,
,1 <jd>
420 draw1,102,8to102,2to104,4to106
,2to106,8:draw1,108,5,108,7 <dn>
430 draw1,92,12to98,12:draw1,95,12
to95,18 <pc>
440 box1,100,10,110,20,,1:box1,110
,10,120,20,,1 <ef>
450 rem =titelschrift setzten===== <aj>
460 gshapeg$(1),128,2:gshapeg$(2),
125,11,2 <cb>
470 fori=2to18step2:draw1,122,ito1
27,i:draw1,313,ito317,i:next:draw1
,319,0to319,20 <ng>
480 x=160:y=100:poke65286,peek(652
86)or16:goto2560 <kd>
490 rem =show modus============== <fp>
500 sshapeic$,i1,u1,i1+10,u1+10:gs
hapeic$,110,0:return <al>
510 rem =draw # 1================ <nl>
520 i1=0:u1=0:mw=1:md$="" <kn>
530 gosub490:poke239,0:rr=0:goto30
80 <fa>
540 rr=1:xx=x:yy=y:goto3080 <cl>
550 rr=0:goto3080:goto540 <fc>
560 rem =draw # 2================ <pi>
570 i1=0:u1=10:mw=2:md$="" <dd>
580 gosub490:poke239,0:rr=0:goto30
80 <mf>
590 rr=2:xx=x:yy=y:goto3080 <ek>
600 rr=0:goto3080:goto590 <bi>
610 rem =box # 1================= <cd>
620 i1=10:u1=0:mw=3:md$="" <io>
630 gosub490:b=0:poke239,0:goto308
0 <ij>
640 x(1)=x:y(1)=y:draw1,x,y <gf>
650 b=1:goto3080 <ak>
660 x(2)=x:y(2)=y <do>
670 box1,x(1),y(1),x(2),y(2) <cp>
680 b=3:goto3080 <hg>
690 rem =box # 2================= <fa>
700 i1=10:u1=10:mw=4:md$="" <co>
710 b=0:gosub490:poke239,0:goto308
0 <jc>
720 x(1)=x:y(1)=y:draw1,x,y <hm>
730 b=1:goto3080 <ol>
740 x(2)=x:y(2)=y <bi>
750 box1,x(1),y(1),x(2),y(2),,1 <bd>
760 b=3:goto3080 <jh>
770 rem =box # 0================= <hg>
780 i1=20:u1=0:mw=5:md$="" <fl>
790 b=0:gosub490:poke239,0:goto308
0 <en>
800 x(1)=x:y(1)=y:draw1,x,y <id>
810 b=1:goto3080 <mi>
820 x(2)=x:y(2)=y <ec>
830 box0,x(1),y(1),x(2),y(2),,1 <he>
```

```
840 b=3:goto3080                        <df>
850 rem =circle==================       <kf>
860 i1=20:u1=10:mw=6:md$=""             <hi>
870 b=0:gosub490:poke239,0:goto308
0                                       <jl>
880 x(1)=x:y(1)=y:draw1,x,y             <io>
890 b=1:goto3080                        <cl>
900 x(2)=x:y(2)=y                       <ll>
910 ifx(2)>=x(1)thenx(2)=x(2)-x(1)
:elsex(2)=x(1)-x(2)                     <gb>
920 ify(2)>=y(1)theny(2)=y(2)-y(1)
:elsey(2)=y(1)-y(2)                     <jj>
930 circle1,x(1),y(1),x(2),y(2):dr
aw0,x(1),y(1)                           <ck>
940 b=3:goto3080                        <ei>
950 rem =rays====================       <mg>
960 i1=40:u1=10:mw=7:md$=""             <hh>
970 b=0:gosub490:poke239,0:goto308
0                                       <df>
980 x(1)=x:y(1)=y:draw1,x,y             <mp>
990 b=1:goto3080                        <if>
1000 x(2)=x:y(2)=y                      <dl>
1010 draw1,x(1),y(1)tox(2),y(2):go
to990                                   <aa>
1020 rem =line==================        <mf>
1030 i1=50:u1=0:mw=8:md$=""             <bn>
1040 b=0:gosub490:poke239,0:goto30
80                                      <ig>
1050 b=1:x(1)=x:y(1)=y:draw1,x,y:g
oto3080                                 <li>
1060 x(2)=x:y(2)=y                      <op>
1070 b=3:draw1,x(1),y(1)tox(2),y(2
):goto3080                              <le>
1080 rem =lines=================        <ia>
1090 i1=50:u1=10:mw=9:md$=""            <el>
1100 b=0:gosub490:poke239,0:goto30
80                                      <ch>
1110 x(1)=x:y(1)=y:draw1,x,y            <bk>
1120 b=1:goto3080                       <ia>
1130 x(2)=x:y(2)=y                      <gm>
1140 draw1,x(1),y(1)tox(2),y(2):x(
1)=x(2):y(1)=y(2):goto1120              <bb>
1150 rem =spray=================        <io>
1160 i1=40:u1=0:mw=10:md$="spray"+
b2$                                     <jj>
1170 b=0:gosub490:poke239,0:goto30
80                                      <nh>
1180 xx=x:yy=y:draw1,x,y                <gn>
1190 onsjgoto3580,3620,3690,3760        <ob>
1200 b=1:goto3080                       <jl>
1210 rem =paint=================        <aj>
1220 i1=30:u1=10:mw=11:md$=""           <hj>
1230 b=0:gosub490:poke239,0:goto30
80                                      <fi>
1240 paint1,x,y                         <jg>
1250 b=1:goto3080                       <hp>
1260 rem =text==================        <mp>
1270 i1=90:u1=10:mw=12:md$=""           <gi>
1280 b=0:gosub490:poke239,0:goto30
80                                      <mo>
1290 poke239,0:cc=0                     <pd>
1300 sshapeap$,int(x/8)*8,int(y/8)
*8,(int(x/8)*8)+8,(int(y/8)*8)+8        <aa>
1310 getw$:char1,int(x/8),int(y/8)
," ",1:ifw$=chr$(13)then1500            <gh>
1320 ifw$=""then1310:elseifw$=chr$
(20)thengshapeap$,int(x/8)*8,int(y
/8)*8:x=x-8:goto1470                    <hh>
1330 ifw$=c3$thengshapeap$,int(x/8
)*8,int(y/8)*8:x=x+8:goto1410           <am>
1340 ifw$=c1$thengshapeap$,int(x/8
)*8,int(y/8)*8:x=x-8:goto1410           <lk>
1350 ifw$=c4$thengshapeap$,int(x/8
)*8,int(y/8)*8:y=y+8:goto1410           <bg>
1360 ifw$=c2$thengshapeap$,int(x/8
)*8,int(y/8)*8:y=y-8:goto1410           <fn>
1370 ifw$=he$thengshapeap$,int(x/8
)*8,int(y/8)*8:x=0:y=25:goto1410        <ml>
1380 ifw$=chr$(18)thenw$="":fr=1:g
oto1410                                 <hf>
1390 ifw$=chr$(146)thenw$="":fr=0:
goto1310                                <cp>
1400 char1,int(x/8),int(y/8),w$,fr
:x=x+8:y=y:cc=cc+1                      <ai>
1410 ifint(x/8)>39thenx=0:y=y+8         <mh>
1420 ifint(y/8)<3theny=168              <ed>
1430 ifint(y/8)>21theny=24              <eg>
1440 ifint(x/8)<0thenx=312              <ic>
1450 goto1300                           <gn>
1460 goto1300                           <ko>
1470 rem =del===================        <pj>
1480 ifint(x/8)<0thenx=312              <fg>
1490 char1,int(x/8),int(y/8)," ":g
oto1300                                 <cl>
1500 gshapeap$,int(x/8)*8,int(y/8)
*8:poke239,0:j=0:goto3080               <ae>
1510 goto3080                           <pm>
1520 rem =load==================        <nh>
1530 poke239,0:char1,1,23,"enter f
ilename:"+b$+b$+b3$                     <kf>
1540 sw$="load":nl$="":zw=0:r=16:c
har1,r,23,ze$                           <fn>
1550 poke239,0                          <cb>
1560 getkeyw$:zw=zw+1                   <pl>
1570 w=asc(w$)                          <jm>
1580 ifw$=chr$(20)andzw>0thenifr>1
6thenr=r-1:zw=zw-2:nl$=left$(nl$,z
w):w$="":goto1760                       <aj>
1590 ifw$=chr$(13)thenchar1,r,23,b
2$:goto1660                             <db>
1600 ifw$=chr$(20)andzw=0then1550       <ok>
1610 ifw$=chr$(32)then1630              <je>
1620 ifw<35orw>129thenifw<149orw>1
91then1550                              <ae>
1630 r=r+1:ifr=36thenr=35               <od>
1640 char1,r-1,23,w$:char1,r,23,ze
```

```
$+" "                                    <fb>
1650 nl$=nl$+w$:goto1550                 <fc>
1660 nl$=left$(nl$,16):gosub3950         <eb>
1670 open15,8,15,"r:"+nl$+"="+nl$:
input#15,d,d$:close15                    <bg>
1680 ifd<>63thenchar1,1,23,"file n
ot found"+b$+b$+b4$                      <np>
1690 poke1319,19:fori=1to4:poke131
9+i,13:nexti:poke239,5                   <gn>
1700 color1,2,5:graphic0:printhe$h
e$cl$"m0"left$(qd$,5)                    <ie>
1710 print"l"+chr$(34)+nl$+chr$(34
)+",8                                    <od>
1720 printc4$c4$"x"                      <di>
1730 printc4$"gR1:color1,1:goto320
0"                                       <lk>
1740 end                                 <cf>
1750 rem =del======================      <dd>
1760 char1,r,23,ze$+" ":goto1560         <lk>
1770 rem =save=====================      <fa>
1780 poke239,0:char1,1,23,"enter f
ilename:"+b$+b$+b3$                      <do>
1790 sw$="save":nl$="":zw=0:r=16:c
har1,r,23,ze$:poke239,0                  <hk>
1800 getkeyw$:zw=zw+1                    <co>
1810 w=asc(w$)                           <ce>
1820 ifw$=chr$(20)andzw>0thenifr>1
6thenr=r-1:zw=zw-2:nl$=left$(nl$,z
w):w$="":goto2000                        <jc>
1830 ifw$=chr$(13)thenchar1,r,23,"
 ":goto1900                              <oc>
1840 ifw$=chr$(20)andzw=0then1800        <ml>
1850 ifw$=chr$(32)then1870               <gl>
1860 ifw<35orw>129thenifw<149orw>1
91then1800                               <oh>
1870 r=r+1:ifr=36thenr=35                <hb>
1880 char1,r-1,23,w$:char1,r,23,ze
$+" "                                    <pb>
1890 nl$=nl$+w$:goto1800                 <gh>
1900 nl$=left$(nl$,16):gosub3950         <lm>
1910 open15,8,15,"r:"+nl$+"="+nl$:
input#15,d,fm$:close15                   <ne>
1920 ifd<>62ord=63thenchar1,1,23,"
file not found"+b$+b$+b4$                <ep>
1930 poke1319,19:fori=1to4:poke131
9+i,13:nexti:poke239,5                   <kh>
1940 color1,2,5:graphic0:printhe$h
e$cl$"m0"left$(qd$,5)                    <pa>
1950 print"s"+chr$(34)+nl$+chr$(34
)+",8,1800,4000"c4$;c4$                  <ag>
1960 print"x"                            <bo>
1970 printc4$"gR1:coL1,1:goto3200"       <pa>
1980 end                                 <ah>
1990 rem =del======================      <fm>
2000 char1,r,23,ze$+" ":goto1820         <gn>
2010 rem =clear====================      <pl>
2020 md$="":char1,1,23,"clear pict
ure : are you sure"+b2$+"y/n"+b5$        <cm>
2030 poke239,0:getkeyw$                  <ge>
2040 ifw$="n"thenchar1,1,23,b$+b3$
:goto2560                                <nl>
2050 ifw$="y"thengraphic1,1:run          <jn>
2060 goto 2030                           <ad>
2070 rem =directory================      <og>
2080 md$="":char1,1,23,"directory
: insert a disk in drive 0"+b2$          <pj>
2090 poke239,0:do:getw$:loopuntilw
$=chr$(32)orjoy(1)=128                   <ck>
2100 printcl$:graphic0:directory:p
rintrn$"press space bar or button"
rf$                                      <pc>
2110 poke239,0:do:getw$:loopuntilw
$=" "orjoy(1)=128:graphic1:goto299
0                                        <mp>
2120 rem =circle 2=================      <im>
2130 i1=30:u1=0:mw=20:md$=""             <kd>
2140 b=0:gosub490:poke239,0:goto30
80                                       <ke>
2150 x(1)=x:y(1)=y:draw1,x,y             <no>
2160 b=1:goto3080                        <bj>
2170 x(2)=x:y(2)=y                       <jn>
2180 ifx(2)>=x(1)thenx(2)=x(2)-x(1
):elsex(2)=x(1)-x(2)                     <bc>
2190 ify(2)>=y(1)theny(2)=y(2)-y(1
):elsey(2)=y(1)-y(2)                     <eo>
2200 circle1,x(1),y(1),x(2),y(2):d
raw0,x(1),y(1)                           <bp>
2210 paint1,x(1),y(1)                    <el>
2220 b=3:goto3080                        <bl>
2230 rem =send disk command=======       <gn>
2240 i1=80:u1=0:gosub490                 <mj>
2250 poke239,0:char1,1,23,"send to
 disk :"+b$+b$+b4$                       <fi>
2260 sd$="":zw=0:r=15:char1,r,23,z
e$:poke239,0                             <gj>
2270 getkeyw$:zw=zw+1                    <hg>
2280 w=asc(w$)                           <la>
2290 ifw$=chr$(20)thenifr>15andzw>
0thenr=r-1:zw=zw-2:sd$=left$(sd$,z
w):w$="":goto2410                        <cf>
2300 ifw$=chr$(13)thenchar1,r,23,b
2$:goto2370                              <dn>
2310 ifw$=chr$(20)andzw=0then2270        <ik>
2320 ifw$=chr$(32)then2340               <hh>
2330 ifw<35orw>129thenifw<149orw>1
91then2270                               <bm>
2340 r=r+1:ifr=36thenr=r-1:zw=zw-1
:goto2270                                <nf>
2350 char1,r-1,23,w$:char1,r,23,ze
$+" "                                    <db>
2360 sd$=sd$+w$:goto2270                 <je>
2370 ifleft$(sd$,4)="read"then2430       <fm>
2380 char1,1,23,"sending disk comm
and"+b$+b5$+b2$:sys65511                 <bd>
2390 open15,8,15,sd$:close15:char1
,1,23,b$+b$+b$+b4$+b3$                   <dm>
```

```
2400 goto2580                              <li>
2410 rem =del===================           <jd>
2420 char1,r,23,ze$+" ":goto2270           <di>
2430 open15,8,15:input#15,f,f$:clo
se15:char1,1,23,"status:"+b$+b$+b5
$+b4$                                      <ph>
2440 char1,8,23,f$                         <no>
2450 do:getw$:loopuntilw$=" "orjoy
(1)>127:goto2580                           <be>
2460 rem =copy==================           <ki>
2470 i1=60:u1=0:mw=19:md$=""               <ab>
2480 b=0:gosub490:poke239,0:goto30
80                                         <lf>
2490 x(1)=x:y(1)=y                         <di>
2500 b=1:goto3080                          <in>
2510 x(2)=x:y(2)=y                         <dg>
2520 sshapesh$,x(1),y(1),x(2),y(2)         <me>
2530 b=3:goto3080                          <pb>
2540 gshapesh$,x,y                         <nd>
2550 b=4:goto3080                          <oj>
2560 rem =menuesteuerung=========          <cb>
2570 xm=60:ym=18                           <bb>
2580 sshapeoh$,xm,ym,xm+5,ym+5:gsh
apepf$,xm,ym,2                             <ng>
2590 char1,1,23,"x = "+str$(x)+b2$
:char1,11,23,"y = "+str$(y)+b$+b4$
+b3$                                       <cl>
2600 j=joy(1):ifj=0theng=2:goto260
0                                          <fk>
2610 gshapeoh$,xm,ym                       <fe>
2620 ifj=128then2750                       <ab>
2630 ifj=5thenym=ym+g                      <gf>
2640 ifj=1thenym=ym-g                      <lm>
2650 ifj=7thenxm=xm-g                      <oo>
2660 ifj=3thenxm=xm+g                      <lb>
2670 ifj=2thenym=ym-g:xm=xm+g              <cd>
2680 ifj=4thenym=ym+g:xm=xm+g              <ja>
2690 ifj=6thenym=ym+g:xm=xm-g              <dk>
2700 ifj=8thenym=ym-g:xm=xm-g              <gn>
2710 ifxm>100thenxm=100:elseifxm<0
thenxm=0                                   <ah>
2720 ifym>20thenym=20:elseifym<0th
enym=0                                     <kb>
2730 sshapeoh$,xm,ym,xm+5,ym+5:gsh
apepf$,xm,ym,2:g=g+2                       <ga>
2740 goto2600                              <fl>
2750 rem =berechnung der position=         <ko>
2760 ifxm>50then2870                       <of>
2770 ifxm>0andxm<10thenifym>0andym
<10then510                                 <jf>
2780 ifxm>0andxm<10thenifym>10andy
m<20then560                                <aa>
2790 ifxm>10andxm<20thenifym>0andy
m<10then610                                <mh>
2800 ifxm>10andxm<20thenifym>10and
ym<20then690                               <mn>
2810 ifxm>20andxm<30thenifym>0andy
m<10then770                                <jj>
2820 ifxm>20andxm<30thenifym>10and
ym<20then850                               <ag>
2830 ifxm>30andxm<40thenifym>0andy
m<10then2120                               <jg>
2840 ifxm>30andxm<40thenifym>10and
ym<20then1210                              <jb>
2850 ifxm>40andxm<50thenifym>0andy
m<10then1150                               <og>
2860 ifxm>40andxm<50thenifym>10and
ym<20then950                               <pj>
2870 ifxm>50andxm<60thenifym>0andy
m<10then1020                               <in>
2880 ifxm>50andxm<60thenifym>10and
ym<20then1080                              <nm>
2890 ifxm>60andxm<70thenifym>0andy
m<10then2460                               <ab>
2900 ifxm>60andxm<70thenifym>10and
ym<20then1520                              <of>
2910 ifxm>70andxm<80thenifym>0andy
m<10then1770                               <ne>
2920 ifxm>70andxm<80thenifym>10and
ym<20then2070                              <bi>
2930 ifxm>80andxm<90thenifym>0andy
m<10then2230                               <dg>
2940 ifxm>80andxm<90thenifym>10and
ym<20then2010                              <og>
2950 ifxm>90andxm<100thenifym>0and
ym<10then3020                              <bk>
2960 ifxm>90andxm<100thenifym>10an
dym<20then1260                             <dk>
2970 goto2580                              <fc>
2980 rem =set freespaces=========          <mj>
2990 char1,1,23,b$+b$+b$+b4$+b4$           <fc>
3000 char1,1,23,"x = "+str$(x)+b2$
:char1,11,23,"y = "+str$(y)+b$+b5$         <ie>
3010 goto2580                              <ef>
3020 rem =reset==================          <ol>
3030 char1,1,23,"reset : are you s
ure y/n"+b$+b3$                            <mo>
3040 poke239,0                             <oh>
3050 getkeyw$:ifw$="y"thengraphic1
,1:printcl$:graphic0:sys65529:else
ifw$<>"n"then3050                          <ch>
3060 char1,1,23,b$+b$+b$+b4$+b3$:g
oto2580                                    <nd>
3070 rem =pfeilsteuerung=========          <ce>
3080 sshapead$,x,y,x+5,y+5:g=1             <pl>
3090 gshapepf$,x,y,2                       <pc>
3100 j=joy(1):p=peek(198):ifj=0and
p=64theng=1:goto3100                       <ck>
3110 gshapead$,x,y                         <ck>
3120 ifp=52then2580                        <cd>
3130 ifj=128thenifmd$<>"spray"+b2$
thensound3,400,5:sound1,410,5:goto
3380                                       <kp>
3140 ifp<>64andmd$="spray"+b2$then
3540                                       <fi>
3150 ifj=5theny=y+g:goto3240               <pf>
```

```
3160 ifj=1theny=y-g:goto3240          <ik>
3170 ifj=7thenx=x-g:goto3240          <gf>
3180 ifj=3thenx=x+g:goto3240          <ed>
3190 ifj=2theny=y-g:x=x+g:goto3240 <ci>
3200 ifj=4theny=y+g:x=x+g:goto3240 <ab>
3210 ifj=6theny=y+g:x=x-g:goto3240 <aa>
3220 ifj=8theny=y-g:x=x-g:goto3240 <bg>
3230 g=1:goto3080                     <mi>
3240 ifpeek(198)=52then2580           <ln>
3250 ifx<0thenx=319                   <cf>
3260 ifx>319thenx=0                   <bl>
3270 ify<21theny=178                  <kk>
3280 ify>178theny=21                  <oc>
3290 ifrr=1thendraw1,xx,yytox,y:xx
=x:yy=y                               <jl>
3300 ifrr=2thendraw0,xx,yytox,y:xx
=x:yy=y                               <lj>
3310 char1,1,23,"x = "+str$(x)+b2$
:char1,11,23,"y = "+str$(y)+b$+b5$ <nk>
3320 sshapead$,x,y,x+5,y+5            <lb>
3330 gshapepf$,x,y,2                  <lg>
3340 g=g+1:goto3090                   <pk>
3350 rem ========================= <oo>
3360 rem  unterprogrammspruenge      <ai>
3370 rem ========================= <cb>
3380 ifmw=1andrr=1then550:elseifmw
=1andrr=0then540                      <fi>
3390 ifmw=2andrr=2then600:elseifmw
=2andrr=0then590                      <hc>
3400 ifmw=3andb=0then640:elseifmw=
3andb=1then660:elseifmw=3andb=3the
n640                                  <je>
3410 ifmw=4andb=0then720:elseifmw=
4andb=1then740:elseifmw=4andb=3the
n720                                  <lg>
3420 ifmw=5andb=0then800:elseifmw=
5andb=1then820:elseifmw=5andb=3the
n800                                  <fc>
3430 ifmw=6andb=0then880:elseifmw=
6andb=1then900:elseifmw=6andb=3the
n880                                  <lj>
3440 ifmw=7andb=0then980:elseifmw=
7andb=1then1000                       <kb>
3450 ifmw=8andb=0then1050:elseifmw
=8andb=1then1060:elseifmw=8andb=3t
hen1050                               <kn>
3460 ifmw=9andb=0then1110:elseifmw
=9andb=1then1130                      <ej>
3470 ifmw=10andb=0then1180:elseifm
w=10andb=1then1190                    <el>
3480 ifmw=11andb=0then1240:elseifm
w=11andb=1then1240                    <hf>
3490 ifmw=12andb=0then1290:elseifm
w=12andb=1then1290                    <mc>
3500 ifmw=19andb=0then2490:elseifm
w=19andb=1then2510:elseifmw=19andb
=3then2540                            <eb>
3510 ifmw=19andb=4then2490            <ig>
3520 ifmw=20andb=0then2150:elseifm
w=20andb=1then2170:elseifmw=20andb
=3then2150                            <jk>
3530 rem =spray part #2=========== <mm>
3540 ifp=56thensj=1:goto1190:elsei
fp=59thensj=2:goto1190                <ld>
3550 ifp=8thensj=3:goto1190:elseif
p=11thensj=4:goto1190                 <kl>
3560 goto 3150                        <dg>
3570 rem =set spray=============== <ei>
3580 draw1,x-3,y-2,x-1,y-1.4,x+1.4
,y-2.3                                <di>
3590 draw1,x-2,y,x-0.5,y+.9,x+2.3,
y+1.1                                 <gi>
3600 draw1,x-2.7,y+1.7,x-.3,y+2.21
,x+2,y+1.7                            <de>
3610 goto1200                         <ho>
3620 draw1,x-5,y-4,x-3,y-3.4,x+3.4
,y-4.2                                <ha>
3630 draw1,x-4,y,x-2.5,y+2.9,x+2.3
,y+3.1,x+4.3,y+.3,x,y                 <je>
3640 draw1,x-4.7,y+3.7,x-2.3,y+4.4
1,x+4,y+4.7                           <im>
3650 draw1,x-3,y-2,x-1,y-1.4,x+1.4
,y-2.3                                <gn>
3660 draw1,x-2,y,x-0.5,y+.9,x+2.3,
y+1.1                                 <gm>
3670 draw1,x-2.7,y+1.7,x-.3,y+2.21
,x+2,y+1.7                            <oh>
3680 goto1200                         <df>
3690 draw1,x-7,y-6,x-5,y-5.4,x+5.4
,y-6.2                                <ca>
3700 draw1,x-6,y,x-4.5,y+4.9,x+4.3
,y+5.1,x+6.3,y-.9,x,y                 <ka>
3710 draw1,x-6.7,y+5.7,x-4.3,y+6.4
1,x+6,y+6.7                           <em>
3720 draw1,x-3,y-2,x-1,y-1.4,x+1.4
,y-2.3                                <ko>
3730 draw1,x-2,y,x-0.5,y+.9,x+2.3,
y+1.1                                 <bp>
3740 draw1,x-2.7,y+1.7,x-.3,y+2.21
,x+2,y+1.7                            <hc>
3750 goto1200                         <aj>
3760 draw1,x-7,y-6,x-5,y-5.4,x+5.4
,y-6.2                                <nk>
3770 draw1,x-6,y,x-4.5,y+4.9,x+4.3
,y+5.1,x+6.3,y+.7,x,y                 <el>
3780 draw1,x-6.7,y+5.7,x-4.3,y+6.4
1,x+6,y+6.7                           <jk>
3790 draw1,x-5,y-4,x-3,y-3.4,x+3.4
,y-4.2                                <jn>
3800 draw1,x-4,y,x-2.5,y+2.9,x+2.3
,y+3.1,x+4.3,y+.3,x,y-.4              <hb>
3810 draw1,x-4.7,y+3.7,x-2.3,y+4.4
1,x,y+5.1,x+.5,y+5.3                  <cm>
3820 draw1,x-3,y-2,x-1,y-1.4,x+1.4
,y-2.3                                <hm>
3830 draw1,x-2,y+1,x-0.5,y+.9,x+2.
```

```
3,y-1.1                                 <ee>
3840 draw1,x-2.7,y+1.7,x-.3,y+3.21
,x+1,y+1.7                              <bn>
3850 draw1,x,y-5,x,y+2,x+5,y+3:got
o1200                                   <lg>
3860 rem =pfeil zeichnen========= <mp>
3870 graphic1,1                         <ap>
3880 draw1,0,0to4,0to0,4to0,0:draw
1,3,0to0,3:draw1,0,0to5,5:paint1,1
,1                                      <mi>
3890 sshapepf$,0,0,5,5:graphic1,1:
graphic0:return                         <aa>
3900 rem =titelbild text========== <pp>
3910 char1,0,0,"welcome to paint s
ystem"                                  <no>
3920 char1,0,1,b2$+"by sascha habe
rlandt"+b2$                             <ka>
3930 sshapeg$(1),0,0,184,7:sshapeg
$(2),0,8,184,16:graphic1,1:return  <fg>
3940 rem =diskettenbefehl========= <ma>
3950 char1,1,23,sw$+" : insert a d
isk into drive 0"+b4$                   <pj>
3960 poke239,0:do:getw$:loopuntilw
$=" "orjoy(1)=128:poke65286,peek(6
5286)and239:return                      <jd>
3970 rem nachspann =============== <dl>
3980 rem * farbcodes/steuercodes * <da>
3990 c4$=chr$(017):rn$=chr$(018)   <ne>
4000 he$=chr$(019):c3$=chr$(029)   <np>
4010 c2$=chr$(145):rf$=chr$(146)   <ie>
4020 cl$=chr$(147):c1$=chr$(157)   <nf>
4030 rem *** zeichensatz/graphik * <dl>
4040 ze$=chr$(175)                      <bk>
4050 rem ********* zeichenfolgen * <fi>
4060 for q=1 to 40                      <pp>
4070 qd$=qd$+c4$                        <li>
4080 next q:b$=chr$(32):b2$=b$+b$  <an>
4090 b3$=b2$+b$:b4$=b3$+b$         <ka>
4100 b5$=b4$+b$:b$=b5$+b5$         <ho>
4110 return                             <mk>
4120 rem ======================== <gb>
4130 rem 60671 bytes memory             <bh>
4140 rem 12777 bytes program            <mp>
4150 rem 00287 bytes variables          <og>
4160 rem 00164 bytes arrays             <jk>
4170 rem 00963 bytes strings            <jf>
4180 rem 12288 bytes graphic            <hf>
4190 rem 34192 bytes free (0)           <di>
4200 rem ======================== <bi>
```


FESTE AUSGABEN

Behalten Sie die Übersicht

Alle festen Ausgaben, die irgendwann einmal fällig werden, können zur besseren Übersicht mit dem Computer erfaßt werden. Falls ein Drucker vorhanden ist, bekommen Sie die Jahresübersicht auch schwarz auf weiß.

Das Programm erstellt einen Überblick über regelmäßig anfallende Ausgaben. Damit sind Unkosten gemeint, die monatlich, vierteljährlich, halbjährlich oder jährlich fällig werden, zum Beispiel Miete, Versicherungen, Vereinsbeiträge.

Das Programm ist vollständig menügesteuert. Jedes Untermenü erlaubt den Rücksprung in das Hauptmenü. Der Abschluß des Programmes ist mit einer Sicherheitsabfrage versehen.

Das Hauptmenü unterteilt sich in die Punkte:

1. Ausgaben auflisten
2. Ausgaben neu eingeben
3. Ausgaben ändern
4. Ausgaben drucken
5. Programmende.

Die Punkte 1 bis 3 führen jeweils in Untermenüs, die grundsätzlich nach der Fälligkeit unterscheiden (monatlich, vierteljährlich, . . .). Die weiteren notwendigen Eingaben werden im Programm ausgewiesen, bedürfen also keiner weiteren Erläuterung.

Punkt 4 ist der interessanteste. Hier wird auf dem angeschlossenen Drucker eine Übersicht in Form einer Tabelle erstellt. Diese Tabelle enthält alle festen Ausgaben, die bis zum Ausdruck abgespeichert wurden. Sie bietet die Möglichkeit, auf einen Blick zu erfassen, in welchem Monat welche Ausgabe (Verwendungszweck und Summe) fällig wird.

Die Druckerroutine von Zeile 3450 bis 3940 paßt für jeden Drucker, da keine spezifischen Steuerzeichen Verwendung fanden. Eventuell empfiehlt es sich für Commodore-Drucker, die OPEN-Anweisung in Zeile 3450 so abzuändern, daß Groß- und Kleinschreibmodus möglich ist. Wer einen breiteren Drucker hat oder ihn auf COMPRESSED PRINT umstellen kann, sollte die Tabelle etwas breiter gestalten, um größere als vierstellige Zahlen und auch Kommastellen zuzulassen.

Die Tabelle kann nur dann richtig erstellt werden, wenn die Fälligkeitsmonate der einzelnen Ausgaben richtig eingegeben werden. Der Vergleich wird anhand der Data am Ende des Listings durchgeführt. Abkürzungen sind hierbei allerdings erlaubt. So kann der Monat Januar mit „Jan" abgekürzt werden.

Damit das Programm einwandfrei arbeiten kann, ist es erforderlich, daß ständig die Arbeitsdiskette mit den abgespeicherten Dateien im Laufwerk ist. Es bietet sich an, Programm und Dateien auf eine Diskette zu nehmen. Die Dateien sind ohnehin nicht sehr umfangreich.

Henning Koglin □

```
10 rem feste ausgaben===========p4 <gn>
20 rem (p) commodore welt team <ho>
30 rem ========================== <ng>
40 rem (c) by henning koglin <ia>
50 rem <pd>
60 rem basic v3.5 <od>
70 rem plus4 (c16/116 + 64 kb) <cd>
80 rem floppy + drucker <gc>
90 rem ========================== <jg>
100 gosub 4000 <eg>
110 rem -------------------------- <cn>
120 rem  einleitung <nk>
130 rem -------------------------- <hh>
140 dimvw$(99),su$(99),mo$(99),mn$
(99),ma$(99),mt$(99),mm$(99),me$(9
9),mq$(99),sm(12) <nm>
150 scnclr:color0,4,1:color4,4,1:c
olor1,4,5 <ja>
160 char1,12,0,rn$+"feste ausgaben
"+rf$:print:print:print:print <mi>
170 print"dieses programm erleicht
ert die ueber-" <pm>
180 print"sicht ueber laufende aus
gaben, die zu" <co>
190 print"bestimmten zeitpunkten f
aellig werden." <ie>
200 print:print"bei der ersten ben
utzung des programmes" <jc>
210 print"muessen zunaechst die er
forderlichen" <jj>
220 print"daten eingegeben werden
!" <ah>
230 char1,9,20,"datum (tt.mm.jj) ?
":inputdd$ <nj>
240 rem -------------------------- <ch>
250 rem  hauptmenue <ma>
260 rem -------------------------- <ob>
270 scnclr:color0,6,1:color4,6,1:c
olor1,6,5 <ja>
280 char1,16,0,rn$+"menue"+rf$:cha
r1,08,5,"1) ausgaben auflisten" <nm>
290 color0,6,1:color4,6,1:color1,6
,5 <eg>
300 char1,08,7,"2) ausgaben neu ei
ngeben":char1,08,9,"3) ausgaben ae
ndern" <fo>
310 char1,08,11,"4) ausgaben druck
en" <lk>
320 char1,8,13,"5) programmende" <fc>
330 char1,4,24,rn$+"bitte 1,2,3,4
oder 5 eingeben !"+rf$ <gl>
340 do:getd$:loopuntild$>"0"andd$<
"6" <eo>
350 b=val(d$):onbgoto430,1240,2240
,3340,360 <lp>
360 rem -------------------------- <gg>
370 rem  programmende <aa>
380 rem -------------------------- <kn>
390 color0,7,3:color4,3,3:color1,2 <ap>
400 scnclr:char1,08,12,rn$+"wirkli
ch beenden "+fl$+"(j/n)"+fo$+" ?" <mo>
410 getkeya$ <fb>
420 ifa$="j"thensys65529:else240 <fd>
430 rem ========================= <im>
440 rem  ausgaben auflisten <am>
450 rem ========================= <do>
460 scnclr:color0,3,1:color4,3,1:c
olor1,3,5 <fp>
470 char1,4,0,rn$+"untermenue 'aus
gaben auflisten'"+rf$ <ak>
480 char1,5,5,"1) monatliche ausga
ben":char1,5,7,"2) vierteljaehrlic
he ausgaben" <oj>
490 char1,5,9,"3) halbjaehrliche a
usgaben":char1,5,11,"4) jaehrliche
 ausgaben" <mi>
500 char1,5,13,"5) monatsuebersich
t" <cb>
510 char1,5,13,"5) zurueck ins hau
ptmenue" <bk>
520 char1,4,24,rn$+"bitte 1,2,3,4
oder 5 eingeben !" <ci>
530 do:getd$:loopuntild$>"0"andd$<
"6" <jg>
540 b=val(d$):onbgoto590,740,910,1
090,550 <cb>
550 rem ------------------------- <ok>
560 rem  zurueck ins hauptmenue <df>
570 rem ------------------------- <nl>
580 goto240 <aj>
590 rem ------------------------- <ag>
600 rem  monatl. auflisten <id>
610 rem ------------------------- <op>
620 scnclr:color0,7,1:color4,7,1:c
olor1,7,5 <jp>
630 open2,8,2,"monatl.,s,r" <ch>
640 input#2,x <mf>
650 fori=0tox:input#2,vw$(i):input
#2,su$(i):next <fe>
660 close2 <mn>
670 char1,09,0,rn$+"monatliche aus
gaben:"+rf$:print:print:i=0:z=0 <aj>
680 do:printvw$(i)tab(30)su$(i) <km>
690 i=i+1:z=z+1:forw=0to39:print"-
";:next <oo>
700 loopuntilz>8ori>x <pf>
710 ifi>xthenchar1,3,24,rn$+"ende
der liste! weiter mit taste!"+rf$:
getkeya$:goto430 <jg>
720 ifi<=xthenchar1,10,24,rn$+"wei
ter mit taste!"+rf$:getkeya$:z=0:s
cnclr <jc>
730 char1,9,0,rn$+"monatliche ausg
aben:"+rf$:print:print:goto680 <hk>
740 rem ------------------------- <bc>
750 rem  vierteljaehrl. auflisten <lk>
```

```
760 rem --------------------------- <jg>
770 scnclr:color0,09,1:color4,09,1
:color1,09,5                        <ko>
780 open2,8,2,"viertelj.,s,r"       <pf>
790 input#2,x                       <co>
800 fori=0tox:input#2,vw$(i),su$(i
),mo$(i),mn$(i),ma$(i),mt$(i)       <op>
810 nexti:close2                    <gk>
820 char1,6,0,rn$+"vierteljaehrlic
he ausgaben:"+rf$:print:print       <ia>
830 i=0:z=0                         <pk>
840 do:printvw$(z)tab(30);rn$;su$(
z);rf$:print"faellig im: ";         <ee>
850 printmo$(z);", ";mn$(z);", ";m
a$(z);", ";mt$(z):i=i+1:z=z+1       <fj>
860 forw=0to39:print"-";:next       <na>
870 loopuntili>5orz>x               <oa>
880 ifz>xthenchar1,3,24,rn$+"ende
der liste! weiter mit taste!"+rf$:
getkeya$:goto430                    <be>
890 ifz<=xthenchar1,10,24,rn$+"wei
ter mit taste!"+rf$:getkeya$:i=0:s
cnclr                               <ah>
900 char1,6,0,rn$+"vierteljaehrlic
he ausgaben:"+rf$:print:print:goto
840                                 <od>
910 rem --------------------------- <fc>
920 rem  halbjaehrl. auflisten      <bg>
930 rem --------------------------- <gp>
940 scnclr:color0,15,3:color4,15,3
:color1,15,6                        <gp>
950 open2,8,2,"halbj.,s,r"          <ip>
960 input#2,x                       <nb>
970 fori=0tox:input#2,vw$(i):input
#2,su$(i):input#2,mm$(i):input#2,m
e$(i):nexti                         <pk>
980 close2                          <na>
990 char1,8,0,rn$+"halbjaehrliche
ausgaben:"+rf$:print:print          <nn>
1000 i=0:z=0                        <bn>
1010 do:printvw$(z)tab(30);rn$;su$
(z);rf$:print"faellig im: ";        <ng>
1020 printmm$(z);", ";me$(z)        <nj>
1030 forw=0to39:print"-";:nextw     <nc>
1040 i=i+1:z=z+1                    <pd>
1050 loopuntili>5orz>x              <hg>
1060 ifz>xthenchar1,3,24,rn$+"ende
 der liste! weiter mit taste!"+rf$
:getkeya$:goto430                   <pj>
1070 ifz<=xthenchar1,10,24,rn$+"we
iter mit taste!"+rf$:getkeya$:i=0:
scnclr                              <lo>
1080 char1,8,0,rn$+"halbjaehrliche
 ausgaben:"+rf$:print:print:goto10
10                                  <le>
1090 rem -------------------------- <kb>
1100 rem  jaehrlich auflisten       <ea>
1110 rem -------------------------- <kh>
1120 scnclr:color0,10,1:color4,10,
1:color1,10,5                       <hl>
1130 open2,8,2,"jaehrl.,s,r"        <fb>
1140 input#2,x                      <bi>
1150 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i):input#2,mq$(i):nexti:cl
ose2                                <gh>
1160 char1,10,0,rn$+"jaehrliche au
sgaben:"+rf$:print:print:i=0:z=0    <pc>
1170 do:printvw$(i)tab(30);rn$;su$
(i);rf$:print"faellig im ";mq$(i)   <fi>
1180 i=i+1:z=z+1                    <ec>
1190 forw=0to39:print"-";:nextw     <jm>
1200 loopuntilz>5ori>x              <kp>
1210 ifi>xthenchar1,3,24,rn$+"ende
 der liste! weiter mit taste!"+rf$
:getkeya$:goto430                   <kg>
1220 ifi<=xthenchar1,10,24,rn$+"we
iter mit taste!"+rf$:getkeya$:z=0:
scnclr                              <gg>
1230 char1,10,0,rn$+"jaehrliche au
sgaben:"+rf$:print:print:goto1170   <pc>
1240 rem ========================== <jp>
1250 rem  ausgaben neu eingeben     <bl>
1260 rem ========================== <gm>
1270 scnclr:color0,8,1:color4,8,1:
color1,8,5                          <jj>
1280 char1,3,0,rn$+"untermenue 'au
sgaben neu eingeben'"+rf$           <ik>
1290 char1,5,5,"1) monatliche ausg
aben":char1,5,7,"2) vierteljaehrli
che ausgaben"                       <gh>
1300 char1,5,9,"3) halbjaehrliche
ausgaben":char1,5,11,"4) jaehrlich
e ausgaben"                         <fe>
1310 char1,5,13,"5) zurueck ins ha
uptmenue"                           <eh>
1320 char1,5,24,ye$+"bitte 1,2,3,4
 oder 5 eingeben !"                 <pc>
1330 do:getd$:loopuntild$>"0"andd$
<"6"                                <cf>
1340 b=val(d$):onbgoto1350,1540,18
00,2010,2200                        <la>
1350 rem -------------------------- <ig>
1360 rem  monatlich eingeben        <jd>
1370 rem -------------------------- <kl>
1380 x=-1                           <ka>
1390 open2,8,2,"monatl.,s,r"        <mo>
1400 input#2,x                      <gj>
1410 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i):nexti                    <ol>
1420 close2                         <lj>
1430 scnclr:color0,1:color4,1:colo
r1,7,5                              <fj>
1440 char1,0,0,rn$+b5$+b4$+"monatl
iche ausgaben"+b$+rf$               <oo>
1450 x=x+1                          <pa>
1460 char1,0,5,"hoehe der ausgabe
```

```
: ":inputsu$(x)                      <gl>
1470 char1,0,7,"verwendungszweck"+
b2$+": ":inputvw$(x)                 <oj>
1480 char1,09,24,"weiter eingeben
(j/n) ?"                             <dm>
1490 getkeya$:ifa$="j"then1430:els
e1500                                <en>
1500 open2,8,2,"@:monatl.,s,w"       <dp>
1510 print#2,x                       <pa>
1520 fori=0tox:print#2,vw$(i):prin
t#2,su$(i):nexti                     <fi>
1530 close2:goto1240                 <ag>
1540 rem ------------------------    <oe>
1550 rem  vierteljaehrl. eingeben    <eh>
1560 rem ------------------------    <im>
1570 x=-1                            <je>
1580 open2,8,2,"viertelj.,s,r"       <pk>
1590 input#2,x                       <fn>
1600 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i)                           <ia>
1610 input#2,mo$(i):input#2,mn$(i)
:input#2,ma$(i):input#2,mt$(i)       <fe>
1620 nexti                           <lf>
1630 close2                          <ac>
1640 scnclr:color0,1:color4,1:colo
r1,3,5                               <cp>
1650 char1,0,0,rn$+b6$+" vierteljа
ehrliche ausgaben "+b6$+rf$          <pa>
1660 x=x+1                           <ef>
1670 char1,0,5,"hoehe der ausgabe
: ":inputsu$(x)                      <il>
1680 char1,0,7,"verwendungszweck"+
b2$+": ":inputvw$(x)                 <dd>
1690 char1,0,9,"faelligkeitsmonate
:":char1,8,11,"1. monat : ":inputm
o$(x)                                <al>
1700 char1,8,12,"2. monat : ":inpu
tmn$(x)                              <in>
1710 char1,8,13,"3. monat : ":inpu
tma$(x)                              <on>
1720 char1,8,14,"4. monat : ":inpu
tmt$(x)                              <ob>
1730 char1,09,24,"weiter eingeben
(j/n) ?"                             <ma>
1740 getkeya$:ifa$="j"then1640:els
e1750                                <in>
1750 open2,8,2,"@:viertelj.,s,w"     <dm>
1760 print#2,x                       <kc>
1770 fori=0tox:print#2,vw$(i):prin
t#2,su$(i):print#2,mo$(i):print#2,
mn$(i)                               <an>
1780 print#2,ma$(i):print#2,mt$(i)
:nexti                               <cl>
1790 close2:goto1240                 <jl>
1800 rem ------------------------    <ii>
1810 rem  halbjaehrlich eingeben     <ea>
1820 rem ------------------------    <hi>
1830 x=-1                            <oe>
1840 open2,8,2,"halbj.,s,r"          <ld>
1850 input#2,x                       <kj>
1860 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i):input#2,mm$(i):input#2,
me$(i):nexti                         <oe>
1870 close2                          <mg>
1880 scnclr:color0,1:color4,1:colo
r1,6,4                               <ih>
1890 char1,0,0,rn$++b4$+b4$+"halbj
aehrliche ausgaben"+b5$+b4$+rf$      <cd>
1900 x=x+1                           <fh>
1910 char1,0,5,"hoehe der ausgabe
: ":inputsu$(x)                      <eb>
1920 char1,0,7,"verwendungszweck"+
b2$+": ":inputvw$(x)                 <jn>
1930 char1,0,9,"faelligkeitsmonate
:":char1,8,11,"1. monat : ":inputm
m$(x)                                <nd>
1940 char1,8,12,"2. monat : ":inpu
tme$(x)                              <ej>
1950 char1,9,24,"weiter eingeben (
j/n) ?"                              <pe>
1960 getkeya$:ifa$="j"then1880:els
e1970                                <pk>
1970 open2,8,2,"@:halbj.,s,w"        <af>
1980 print#2,x                       <ha>
1990 fori=0tox:print#2,vw$(i):prin
t#2,su$(i):print#2,mm$(i):print#2,
me$(i):nexti                         <gl>
2000 close2:goto1240                 <oo>
2010 rem ------------------------    <nb>
2020 rem  jaehrlich eingeben         <fi>
2030 rem ------------------------    <ga>
2040 x=-1                            <ah>
2050 open2,8,2,"jaehrl.,s,r"         <gn>
2060 input#2,x                       <no>
2070 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i):input#2,mq$(i):nexti:cl
ose2                                 <ne>
2080 scnclr:color0,1:color4,1:colo
r1,15,5                              <ec>
2090 char1,0,0,rn$+b5$+b4$+"jaehrl
iche ausgaben "+b$+rf$               <fc>
2100 x=x+1                           <ia>
2110 char1,0,5,"hoehe der ausgabe
: ":inputsu$(x)                      <bk>
2120 char1,0,7,"verwendungszweck"+
b2$+": ":inputvw$(x)                 <in>
2130 char1,0,9,"faelligkeitsmonat
: ":inputmq$(x)                      <km>
2140 char1,9,24,"weiter eingeben (
j/n) ?"                              <bp>
2150 getkeya$:ifa$="j"then2080:els
e2160                                <jb>
2160 open2,8,2,"@:jaehrl.,s,w"       <lf>
2170 print#2,x                       <ge>
2180 fori=0tox:print#2,vw$(i):prin
t#2,su$(i):print#2,mq$(i):nexti:cl
```

```
ose2                                   <eh>
2190 goto1240                          <fe>
2200 rem -------------------------     <mo>
2210 rem  zurueck ins hauptmenue       <of>
2220 rem -------------------------     <fb>
2230 goto240                           <jb>
2240 rem =========================     <ml>
2250 rem  ausgaben aendern             <dh>
2260 rem =========================     <jp>
2270 scnclr:color0,4,4:color4,4,4:
color1,4,1                             <ki>
2280 char1,5,0,rn$+"untermenue 'au
sgaben aendern'"+rf$                   <pf>
2290 char1,5,5,"1) monatliche ausg
aben":char1,5,7,"2) vierteljaehrli
che ausgaben"                          <ig>
2300 char1,5,9,"3) halbjaehrliche
ausgaben":char1,5,11,"4) jaehrlich
e ausgaben"                            <lf>
2310 char1,5,13,"5) zurueck ins ha
uptmenue"                              <om>
2320 char1,6,24,rn$+"bitte1,2,3 od
er 4 eingeben !"+rf$                   <op>
2330 do:getd$:loopuntild$>"0"andd$
<"6"                                   <jd>
2340 b=val(d$):onbgoto2390,2620,28
80,3120,2350                           <oe>
2350 rem -------------------------     <ml>
2360 rem  zurueck ins hauptmenue       <ip>
2370 rem -------------------------     <ig>
2380 goto240                           <pk>
2390 rem -------------------------     <kl>
2400 rem  monatlich aendern            <ni>
2410 rem -------------------------     <gl>
2420 scnclr:color0,3,4:color4,3,4:
color1,3,1                             <il>
2430 char1,0,0,rn$+b6$+" monatlich
e ausgaben aendern"+b6$+rf$:print:
print                                  <jk>
2440 open2,8,2,"monatl.,s,r"           <ia>
2450 input#2,x                         <fg>
2460 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i):nexti :close2               <ok>
2470 fori=0tox:scnclr:char1,0,0,rn
$+b6$+" monatliche ausgaben aender
n"+b6$+rf$                             <pg>
2480 char1,0,5,"hoehe der ausgabe
: ":printsu$(i)                        <fm>
2490 char1,0,7,"verwendungszweck"+
b2$+": ":printvw$(i)                   <nm>
2500 char1,0,20,rn$+b6$+"a=aendern
"+b3$+"w=weiter"+b3$+"e=ende"+b5$+
rf$                                    <ij>
2510 getkeya$                          <am>
2520 ifa$="a"thengoto2570              <jp>
2530 ifa$="w"then2550                  <ke>
2540 ifa$="e"then2580:else2580         <lg>
2550 ifi=xthen2580                     <ph>
2560 nexti                             <hg>
2570 char1,18,5,"":inputsu$(i):cha
r1,18,7,"":inputvw$(i):goto2550        <fc>
2580 open2,8,2,"@:monatl.,s,w"         <cc>
2590 print#2,x                         <lm>
2600 fori=0tox:print#2,vw$(i):prin
t#2,su$(i):nexti:close2                <fm>
2610 goto2240                          <ac>
2620 rem -------------------------     <kb>
2630 rem  vierteljaehrl. aendern       <od>
2640 rem -------------------------     <kh>
2650 scnclr:color0,6,4:color4,6,4:
color1,6,1                             <ob>
2660 open2,8,2,"viertelj.,s,r"         <nb>
2670 input#2,x                         <bi>
2680 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i):input#2,mo$(i):input#2,
mn$(i)                                 <ne>
2690 input#2,ma$(i):input#2,mt$(i)
:nexti:close2                          <bh>
2700 fori=0tox:scnclr:char1,0,0,rn
$+b3$+"vierteljaehrliche ausgaben
aendern"+b3$+rf$                       <fn>
2710 char1,0,5,"hoehe der ausgabe
: ":printsu$(i)                        <cl>
2720 char1,0,7,"verwendungszweck"+
b2$+": ":printvw$(i):char1,0,9,"fa
elligkeitsmonate:"                     <in>
2730 char1,8,11,"1. monat : ":prin
tmo$(i):char1,8,12,"2. monat : ":p
rintmn$(i)                             <hn>
2740 char1,8,13,"3. monat : ":prin
tma$(i):char1,8,14,"4. monat : ":p
rintmt$(i)                             <df>
2750 char1,0,20,rn$+b6$+"a=aendern
"+b3$+"w=weiter"+b3$+"e=ende"+b5$+
rf$:getkeya$                           <ae>
2760 ifa$="a"then2800                  <jd>
2770 ifa$="w"then2780:else2830         <nm>
2780 ifi=xthen2830                     <op>
2790 nexti                             <ap>
2800 char1,18,5,"":inputsu$(i):cha
r1,18,7,"":inputvw$(i)                 <hd>
2810 char1,17,11,"":inputmo$(i):ch
ar1,17,12,"":inputmn$(i)               <fh>
2820 char1,17,13,"":inputma$(i):ch
ar1,17,14,"":inputmt$(i):goto2780      <ed>
2830 open2,8,2,"@:viertelj.,s,w"       <gg>
2840 print#2,x                         <go>
2850 fori=0tox:print#2,vw$(i):prin
t#2,su$(i):print#2,mo$(i):print#2,
mn$(i)                                 <dd>
2860 print#2,ma$(i):print#2,mt$(i)
:nexti:close2                          <ml>
2870 goto2240                          <kd>
2880 rem -------------------------     <ig>
2890 rem  halbjaehrl. aendern          <ce>
2900 rem -------------------------     <kl>
```

```
2910 scnclr:color0,5,4:color4,5,4:
color1,5,1                          <ji>
2920 open2,8,2,"halbj.,s,r"         <nj>
2930 input#2,x                      <gj>
2940 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i):input#2,mm$(i):input#2,
me$(i)                              <ne>
2950 nexti:close2                   <po>
2960 fori=0tox:scnclr:char1,0,0,rn
$+b5$+"halbjaehrliche ausgaben aen
dern"+b4$+rf$                       <eh>
2970 char1,0,5,"hoehe der ausgabe
: ":printsu$(i)                     <ik>
2980 char1,0,7,"verwendungszweck"+
b2$+": ":printvw$(i):char1,0,9,"fa
elligkeitsmonate:"                  <lo>
2990 char1,8,11,"1. monat : ":prin
tmm$(i):char1,8,12,"2. monat : ":p
rintme$(i)                          <ad>
3000 char1,0,20,rn$+b6$+"a=aendern
"+b3$+"w=weiter"+b3$+"e=ende"+b5$+
rf$:getkeya$                        <fl>
3010 ifa$="a"then3050               <df>
3020 ifa$="w"then3030:else3070      <ca>
3030 ifi=xthen3070                  <kn>
3040 nexti                          <pl>
3050 char1,18,5,"":inputsu$(i):cha
r1,18,7,"":inputvw$(i)              <kl>
3060 char1,17,11,"":inputmm$(i):ch
ar1,17,12,"":inputme$(i):goto3030   <ip>
3070 open2,8,2,"@:halbj.,s,w"       <nd>
3080 print#2,x                      <hp>
3090 fori=0tox:print#2,vw$(i):prin
t#2,su$(i):print#2,mm$(i):print#2,
me$(i)                              <lp>
3100 nexti:close2                   <mp>
3110 goto2240                       <mj>
3120 rem ------------------------   <fh>
3130 rem  jaehrlich aendern         <da>
3140 rem ------------------------   <om>
3150 scnclr:color0,12,4:color4,12,
4:color1,12,1                       <kp>
3160 open2,8,2,"jaehrl.,s,r"        <ji>
3170 input#2,x                      <hk>
3180 fori=0tox:input#2,vw$(i):inpu
t#2,su$(i):input#2,mq$(i):nexti:cl
ose2                                <kk>
3190 fori=0tox:scnclr:char1,0,0,rn
$+b6$+"jaehrliche ausgaben aendern
 "+b6$+rf$                          <em>
3200 char1,0,7,"hoehe der ausgabe
: ":printsu$(i)                     <ld>
3210 char1,0,9,"verwendungszweck"+
b2$+": ":printvw$(i)                <ok>
3220 char1,0,11,"faelligkeitsmonat
 : ":printmq$(i)                    <gb>
3230 char1,0,20,rn$+b6$+"a=aendern
"+b3$+"w=weiter"+b3$+"e=ende"+b5$+
rf$:getkeya$                        <og>
3240 ifa$="a"then3280               <hf>
3250 ifa$="w"then3260:else3300      <ho>
3260 ifi=xthen3300                  <nj>
3270 nexti                          <jh>
3280 char1,18,7,"":inputsu$(i):cha
r1,18,9,"":inputvw$(i):char1,18,11
,"":inputmq$(i)                     <pg>
3290 goto3260                       <gc>
3300 open2,8,2,"@:jaehrl.,s,w"      <go>
3310 print#2,x                      <of>
3320 fori=0tox:print#2,vw$(i):prin
t#2,su$(i):print#2,mq$(i):nexti:cl
ose2                                <jm>
3330 goto2240                       <gn>
3340 rem ======================== <ma>
3350 rem  ausgaben drucken          <nk>
3360 rem ======================== <di>
3370 scnclr:color0,13,4:color4,13,
4:color1,13,1                       <gn>
3380 char1,0,0,rn$+b$+b2$+"ausgabe
n drucken"+b$+b2$+rf$:char1,0,3,""  <ln>
3390 print"mit diesem unterprogram
m wird eine "                       <la>
3400 print"uebersicht aller ausgab
en ausgedruckt."                    <ai>
3410 print:print"bitte gedulden si
e sich etwas, der aus-"             <ja>
3420 print"druck dauert einige zei
t!!"                                <el>
3430 char1,9,24,rn$+"bitte taste d
ruecken !"+rf$:getkeya$             <km>
3440 rem ----kopfzeile drucken---- <hg>
3450 open4,4                        <nc>
3460 print#4,"uebersicht ueber fes
te ausgaben";:print#4,b3$"datum: "
;dd$:print#4                        <eb>
3470 print#4,"verwendungszweck *";  <im>
3480 restore3980:fori=1to12:readx$
:print#4,"* "left$(x$,3);:next:pri
nt#4                                <pn>
3490 forza=1to78:print#4,"*";:next
za:print#4                          <od>
3500 rem ----monatl. ausgaben----- <ih>
3510 open2,8,2,"monatl.,s,r":pp=0   <on>
3520 forg=1to12:sm(g)=0:next        <ch>
3530 input#2,x                      <bc>
3540 fori=0tox:input#2,vw$,su:prin
t#4,left$(vw$+b$+b6$,16)" *";       <pi>
3550 forg=1to12:print#4,"*"using"#
###";su;:sm(g)=sm(g)+su:next:print
#4                                  <ne>
3560 nexti:close2                   <hf>
3570 rem ----viertelj. ausgaben--- <lm>
3580 open2,8,2,"viertelj.,s,r":pp=
0                                   <il>
3590 input#2,x                      <no>
3600 fori=0tox:input#2,vw$,su,mo$,
```

```
mn$,ma$,mt$                          <jh>
3610 print#4,left$(vw$+b$+b6$,16)"
 *";                                 <cd>
3620 restore3980:forg=1to12:readmz
$                                    <mp>
3630 fl=left$(mz$,len(mo$))=mo$      <pj>
3640 fl=florleft$(mz$,len(mn$))=mn
$                                    <ak>
3650 fl=florleft$(mz$,len(ma$))=ma
$                                    <hd>
3660 fl=florleft$(mz$,len(mt$))=mt
$                                    <co>
3670 ifflthenprint#4,"*"using"####
";su;:sm(g)=sm(g)+su:elseprint#4,"
*"b4$;                               <ng>
3680 nextg:print#4                   <ge>
3690 nexti:close2                    <mf>
3700 rem ----halbj. ausgaben------   <le>
3710 open2,8,2,"halbj.,s,r"          <ej>
3720 input#2,x                       <ag>
3730 fori=0tox:input#2,vw$,su,mm$,
mq$                                  <ok>
3740 print#4,left$(vw$+b$+b6$,16)"
 *";                                 <od>
3750 restore3980:forg=1to12:readmz
$                                    <af>
3760 fl=left$(mz$,len(mm$))=mm$      <fn>
3770 fl=florleft$(mz$,len(mq$))=mq
$                                    <fe>
3780 ifflthenprint#4,"*"using"####
";su;:sm(g)=sm(g)+su:elseprint#4,"
*"b4$;                               <nb>
3790 nextg:print#4                   <dk>
3800 nexti:close2                    <jh>
3810 rem ----jaehrliche ausgaben--   <fl>
3820 open2,8,2,"jaehrl.,s,r"         <dg>
3830 input#2,x                       <nn>
3840 fori=0tox:input#2,vw$,su,mq$    <ie>
3850 print#4,left$(vw$+b$+b6$,16)"
 *";                                 <fl>
3860 restore3980:forg=1to12:readmz
$                                    <cp>
3870 fl=left$(mz$,len(mq$))=mq$      <bd>
3880 ifflthenprint#4,"*"using"####
";su;:sm(g)=sm(g)+su:elseprint#4,"
*"b4$;                               <af>
3890 pk=pk+1:nextg:print#4           <cn>
3900 nexti:close2                    <cd>
3910 rem ----monatliche summen----   <eb>
3920 forza=1to78:print#4,"*";:next
:print#4                             <jf>
3930 print#4,"monatliche summe *";   <ig>
3940 forg=1to12:print#4,"*"using"#
###";sm(g);:next:print#4:goto240     <np>
3950 rem ------------------------    <ea>
3960 rem  datas                      <ei>
3970 rem ------------------------    <jc>
3980 datajanuar,februar,maerz,apri
l,mai,juni,juli,august,september,o
ktober                               <og>
3990 datanovember,dezember           <cp>
4000 rem nachspann ==============    <nm>
4010 rem * farbcodes/steuercodes *   <mb>
4020 rn$=chr$(018):fl$=chr$(130)     <ef>
4030 fo$=chr$(132):rf$=chr$(146)     <bf>
4040 ye$=chr$(158)                   <bo>
4050 b$=chr$(32):b2$=b$+b$           <pj>
4060 b3$=b2$+b$:b4$=b3$+b$           <ij>
4070 b5$=b4$+b$:b6$=b5$+b$           <he>
4080 b$=b5$+b$:return                <ma>
4090 rem ========================    <gf>
4100 rem 60671 bytes memory          <gb>
4110 rem 13530 bytes program         <cb>
4120 rem 00217 bytes variables       <ln>
4130 rem 02835 bytes arrays          <nh>
4140 rem 00608 bytes strings         <bn>
4150 rem 43481 bytes free (0)        <pm>
4160 rem ========================    <pd>
```


DER COMPUTER SPIELT NACH NOTEN

Haste Töne?

Vom Blatt zu singen oder zu spielen, ist nicht jedermanns Sache. Schreiben Sie die Noten doch einfach ab, und zwar mit dem Computer. Dann können Sie hören, wie es richtig klingt. Außerdem läßt sich die Melodie in eigene Programme einbinden. Gleich zwei Programme sorgen für die richtige Musik.

Beide Programme gehören zusammen. Das erste Programm heißt „Notenblatt", das zweite „Datamaker Musik".

NOTENBLATT

Nach dem Start können Erklärungen abgerufen werden. Danach wird das Notenblatt mit vier Notenzeilen und Tastenerklärungen gezeichnet.
Jeder, der mit der Musik nicht allzuviel am Hut hat, der auch nicht Klavierspielen kann und somit mit

professionellen Programmen nicht zurechtkommt, kann mit diesem Programm die Noten aus einem Liederbuch einfach abschreiben.
Das Programm verarbeitet 18 Noten mit den dazugehörigen Halbtönen. Eingangs gesetzte # oder b werden über das ganze Stück beibehalten. Auflösungen von # oder b im Takt werden nach Setzen des nächsten Taktstriches für weitere Eingaben wieder aufgehoben. Ebenso gelten im Takt geschriebene # oder b nur für den betreffenden Takt.
Es ist auch möglich, die im Stück vorgesehenen Pausen mit einzugeben.
Das Stück kann jederzeit abgespielt werden, danach lassen sich weitere Eingaben vornehmen. Insgesamt sind 500 Töne und Pausen möglich.
Das Stück kann nach Beendigung der Eingabe auf Diskette gespeichert werden.

DATAMAKER MUSIK

Dieses Programm wandelt ein sequentielles File (Musikstück) in Datazeilen zur Verwendung in eigenen Programmen um. Um diese Data auch verwenden zu können, ist es am besten, das in den Zeilen 500 bis 600 des Programms unter REM angegebene Programm zu verwenden, das auch einen Ausstieg aus dem Musikstück ermöglicht.

1. Notenblatt

Dokumentation:

Zeilen 100– 590	Notenblatt und SHAPES zeichnen.
Zeilen 610– 810	Hauptteil der Eingabe.
Zeilen 830– 900	Erkennen der Tonhöhe und Tondauer.
Zeilen 920–1020	Unterprogramm "Ende".
Zeilen 1040–1160	Laden einer Melodie von Diskette.
Zeilen 1180–1310	Speichern einer Melodie (Diskette).
Zeilen 1320–1520	Setzen der Tonerhöhungs- (#) und Tonerniedrigungszeichen (b).
Zeilen 1540–1580	Spielen einer eingegebenen oder geladenen Melodie.
Zeilen 1600–1660	Schreiben des Auflösungszeichens (# und b werden für einen Takt aufgehoben).
Zeilen 1680–1800	Setzen des Taktstriches.
Zeilen 1820–1870	Notenzahl (500 Noten) überschritten.
Zeilen 1890–2030	Löschen einer Eingabe. Nur die jeweils letzte Eingabe kann gelöscht werden.
Zeilen 2050–2160	Schreiben des Wiederholungszeichens. Die Noten, die zwischen zwei Wiederholungszeichen stehen, werden nochmals übernommen.
Zeilen 2180–2260	Setzen des Eingabepfeiles.
Zeilen 2280–2340	Schreiben der Pausen.
Zeilen 2360–2860	Variablen, Einleitung.
Zeilen 2870–2980	Fehlerkanalabfrage.
Zeilen 3000–3170	Notendata.

Erläuterungen zum Programmablauf:

Nachdem das Programm mit RUN gestartet wurde, können Bedienungsinstruktionen abgefragt werden. Danach wird das Notenblatt gezeichnet sowie am unteren Bildschirmrand die Tastenbedeutung angezeigt. Im einzelnen:

0 = Wiederholung Anfang.
g = Wiederholung Ende.
1 = Ganze Note, 2 = halbe Note, 3 = viertel Note, 4 = achtel Note, 5 = sechzehntel Note, 6 = zweiunddreißigstel Note.
7 = #, 8 = b.
A = Auflösungszeichen, hebt # oder b für diesen Takt auf.
B = Zeichnet den Taktstrich (wichtig, wenn innerhalb eines Takts ein Auflösungszeichen oder ein # oder ein b geschrieben wurde).
C = Ganze Pause, D = halbe Pause, E = viertel Pause.
F = Achtel Pause, G = sechzehntel Pause.
H = Ende des Programms mit Sicherheitsabfrage, ob Melodie gespeichert wurde.
L = Laden einer Melodie von Diskette.
S = Speichern einer Melodie auf Diskette.
P = Spielen einer eingegebenen Melodie zu jedem Zeitpunkt der Eingabe oder einer von Diskette geladenen Melodie.
N = Löschen aller Eingaben und Neustart. Vorsicht: Keine Sicherheitsabfrage!
+ = Tondauer des zuletzt eingegebenen Tones oder einer Pause wird um die Hälfte des Wertes verlängert.
– = Löscht die letzte Eingabe (Note oder Taktstrich oder # oder b oder Auflösungszeichen oder Pause oder Wiederholungszeichen).

Zur Eingabe:

Nachdem das Notenblatt und der Eingabepfeil oben links gezeichnet sind, können die Noten von einem Musikstück abgeschrieben werden.
Zu Anfang – wie beim Musikstück auch – wird die Tonart durch # oder b bestimmt. Dazu mit „Cursor down" und „Cursor up" auf oder zwischen die entsprechenden Linien der ersten Notenzeile fahren und 7 oder 8 drücken. Danach auf die entsprechende Notenlinie fahren und das nächste # oder b schreiben. Diese Eingaben gelten für das ganze Stück.
Jede Notenzeile besteht aus fünf Notenlinien und vier Hilfslinien. Die Hilfslinien sind auf einem normalen Notenblatt nicht gezeichnet und hier nur gestrichelt. Im Bereich der unteren Hilfslinie bis zur oberen Hilfslinie jeder Notenzeile können Noten plaziert werden. Kommt im Musikstück ein Taktstrich, ist dieser mit B zu schreiben. Die richtige Spalte sucht sich der Computer selbst für alle Eingaben. Die Wiederholungszeichen sehen wie folgt aus:
Anfang I I : 0
Ende : I I g
Wenn im Musikstück von Anfang bis Ende wiederholt werden soll, genügt die Eingabe g am Ende des Musikstückes, und das ganze Stück wird wiederholt (nicht nochmal geschrieben, aber die Noten werden zweimal hintereinander gespeichert).
Kommt im Musikstück zwischendurch ein # oder b, so ist unbedingt auf die Eingabe des Taktstriches zu achten, denn # oder b gelten nur bis zum nächsten Taktstrich. Das gleiche gilt für ein Auflösungszeichen. Wird der Taktstrich vergessen, wird ab dieser Stelle das ganze Stück falsch, da dann jede entsprechende Note mit Erhöhung, Erniedrigung oder beim Auflösungszeichen ohne diese gespeichert wird.
Während der Eingabe kann jederzeit das Musikstück mit P gespielt und die Eingabe dann fortgesetzt werden.

Lesen Sie bitte weiter auf Seite 113

```
10 rem notenblatt =============p4 <li>
20 rem (p) commodore welt team    <ho>
30 rem ========================== <ng>
40 rem (c) by reiner hickel       <hg>
50 rem                            <pd>
60 rem                            <ah>
70 rem basic v3.5                 <nl>
80 rem plus4 (c16/116 + 64kb),disk <gc>
90 rem ========================== <jg>
100 graphic1,1:clr:gosub3250:color
1,2,4:color0,1:scnclr             <gl>
110 gosub2360:volla               <kn>
120 scnclr:forj=16to136step40     <mk>
130 fori=0to4:sound3,750+i*50,5   <lm>
140 draw1,10,j+i*4to319,j+i*4     <ng>
150 nexti,j                       <ee>
160 forj=8to128step40             <jg>
170 fori=jtoj+4step4              <dp>
180 fork=10to314step8:sound3,700+k
,2                                <em>
190 draw1,k,itok+4,i              <go>
200 draw1,k,i+28tok+4,i+28        <no>
210 nextk,i,j                     <lo>
220 draw1,10,0to10,168:sound3,900,
1                                 <ln>
230 draw1,0,168to319,168to319,169t
o0,169:sound3,900,2               <dg>
240 char,11,22,"tastenbedeutungen" <ic>
250 char1,0,22,"loesch= -":char1,0
,23,"play"+b2$+"= p"              <fp>
260 char1,0,24,"save"+b2$+"= s"   <mk>
270 char1,30,22,"punkt = +":char1,
31,23,"ende = h"                  <og>
280 char1,31,24,"load = l"        <jb>
290 z=0:fori=11to20               <pa>
300 char,11,23,"0123456789abcdefg" <kh>
310 draw1,89,192to89,198:draw1,91,
192to91,198:draw1,93,194:draw1,93,
196:sound3,900,4                  <mo>
320 fori=99to139step8:sound3,850+i
,45                               <jh>
330 circle1,i,197,2,1             <id>
340 ifi>107thenpaint1,i,197,1:soun
d3,650,5                          <aa>
350 nexti                         <mj>
360 fori=109to141step8:sound3,850+
i,1                               <io>
370 draw1,i,192toi,197            <hg>
380 nexti:sound3,950,10           <cc>
390 draw1,125,192to127,192        <dp>
400 draw1,133,192to135,192:draw1,1
33,193to135,193                   <nh>
410 draw1,141,192to143,192:draw1,1
41,193to143,193:draw1,141,194to143
,194                              <of>
420 draw1,146,194to146,199:draw1,1
48,193to148,198:draw1,145,195to150
,192                              <pj>
430 draw1,145,199to150,196        <hk>
440 draw1,154,192to154,198to157,19
6to157,194to154,194               <og>
450 draw1,161,194:draw1,161,196:dr
aw1,163,192to163,198:draw1,165,192
to165,198                         <fg>
460 draw1,169,192to169,197to172,19
6                                 <oi>
470 draw1,169,194to172,193to172,19
9                                 <ad>
480 draw1,179,192to179,199        <oh>
490 draw1,186,193to189,193to189,19
4to186,194:draw1,184,192to191,192 <nc>
500 draw1,192,195to199,195:draw1,1
94,193to197,193to197,194to194,194 <ek>
510 draw1,203,192to204,194to203,19
6to204,197to203,198               <mk>
520 draw1,210,194to214,194to211,19
9                                 <oh>
530 draw1,218,194to222,194to217,19
9:draw1,218,195to221,195          <ge>
540 draw1,3,6to6,8to3,10:ifoo=1the
noo=0:ns=an+10:z=1:zp=6:return    <df>
550 fori=88to216step8             <dj>
560 sshapen$((i-88)/8),i,192,i+7,1
99                                <pi>
570 nexti:sound1,500,3            <jg>
580 sshapep$,0,6,6,10:sshapecl$,0,
11,6,19                           <em>
590 sshapecn$,11,0,18,42          <jh>
600 rem =hauptschleife=========== <lm>
610 z=1                           <bo>
620 do                            <mh>
630 getq$:ifq$=""then810          <hn>
640 sound3,934,1                  <kf>
650 ifq$=cu$thenh=zp:zp=zp+2:gosub
2180                              <mj>
660 ifq$=co$thenh=zp:zp=zp-2:gosub
2180                              <nd>
670 ifq$="0"thenwa=nz:gosub2050   <no>
680 ifq$="9"thenwe=nz:gosub2050   <gd>
690 ifq$="+"thensound3,750,5:l(nz)
=1.5*l(nz)                        <pl>
700 ifq$="-"thengosub1890         <pj>
710 ifq$="8"orq$="7"thengosub1330 <mm>
720 ifq$>"0"andq$<"7"thengosub830 <bk>
730 ifq$="a"thengosub1600         <cn>
740 ifq$="b"thengosub1680         <ib>
750 ifq$="h"then920               <jb>
760 ifq$="l"thengosub1040         <nh>
770 ifq$="s"thengosub1180         <em>
780 ifq$="p"thengosub1540         <io>
790 ifq$="n"thenrun               <jc>
800 ifq$>"b"andq$<"h"thengosub2280 <ka>
810 loop                          <eg>
820 rem =noten erkennen========== <nj>
830 gosub2180:w=(zp-(z-1)*40-4)/2 <lk>
840 a=h(w)+a(w)+1:q=val(q$):h=zp-3 <el>
```

```
850 nz=nz+1:ifnz>500thengosub1820   <oe>
860 s(nz)=t(w,a)                    <ea>
870 l(nz)=64/(2↑(q-1))              <al>
880 sound1,s(nz),l(nz)              <ah>
890 gosub2240                       <gl>
900 lo=0:return                     <kj>
910 rem =ende==================== <ni>
920 graphic0,1                      <pc>
930 printcu$cu$tab(3)"ist die melo
die gespeichert? (j/n)"             <id>
940 getkeyx$                        <lk>
950 ifx$="j"thengraphicclr:end      <ph>
960 printcu$tab(3)"soll sie gespei
chert werden? (j/n)"                <mo>
970 getkeyx$                        <ji>
980 ifx$="j"thengosub1180           <em>
990 printcu$tab(3)"weitere melodie
n eingeben (j/n)"                   <lk>
1000 getkeyx$:ifx$="j"thengraphic1
,1:goto120                          <gm>
1010 graphicclr:end                 <lb>
1020 goto920                        <ln>
1030 rem =melodie laden=========== <ia>
1040 graphic0,1                     <hk>
1050 printcu$cu$tab(3)"mit <d> wer
den alle musiktitel"                <hm>
1060 printcu$tab(3)"ausgedruckt, s
onst andere taste"                  <bc>
1070 getkeyx$:ifx$="d"thendirector
y"*=s"                              <ce>
1080 printcu$tab(3);:input"titel d
er melodie";na$:na$=left$(na$,12)   <pl>
1090 open2,8,2,na$+",s,r"           <an>
1100 input#2,nz                     <hh>
1110 fori=1tonz                     <jo>
1120 input#2,s(i)                   <bc>
1130 input#2,l(i)                   <ml>
1140 nexti                          <cn>
1150 close2                         <hn>
1160 graphic1,0:return              <cc>
1170 rem =melodie speichern======= <ml>
1180 graphic0,1                     <eb>
1190 printcu$cu$tab(3);:input"tite
l der melodie";na$:na$=left$(na$,9
)                                   <ok>
1200 open15,8,15                    <lp>
1210 open2,8,2,na$+",s,w"           <if>
1220 gosub2870:ifu=1thenu=0:goto11
90                                  <jp>
1230 ifu=2thenu=0:goto1200          <ok>
1240 ifu=3thenu=0:return            <nm>
1250 print#2,nz                     <il>
1260 fori=1tonz                     <nl>
1270 print#2,s(i)                   <gb>
1280 print#2,l(i)                   <ga>
1290 nexti                          <ik>
1300 close2:close15                 <gd>
1310 graphic1,0:return              <lc>
1320 rem =setzen von # oder b===== <ni>
1330 ifnz>0then1440                 <on>
1340 w=(zp-(z-1)*40-4)/2:ifval(q$)
=7thenh(w)=1:elseh(w)=-1            <pl>
1350 ifw<5thenh(w+7)=h(w):h(w+14)=
h(w)                                <hn>
1360 ifw>4andw<12thenh(w+7)=h(w)    <cp>
1370 ifw>7andw<15thenh(w-7)=h(w)    <be>
1380 ifw>14thenh(w-14)=h(w):h(w-7)
=h(w)                               <fb>
1390 fori=zp-2to120+zp-2step40      <kb>
1400 gshapen$(val(q$)),ns,i,2       <gm>
1410 nexti                          <gm>
1420 ns=ns+10:an=ns:xy=xy+1         <ad>
1430 lo=0:return                    <le>
1440 gshapen$(val(q$)),ns,zp-2,2:x
y=1                                 <ae>
1450 w=(zp-(z-1)*40-4)/2:ifval(q$)
=7thenh(w)=1:elseh(w)=-1:r=w        <nb>
1460 ifw<5thenh(w+7)=h(w):h(w+14)=
h(w):q=w+7:t=w+14                   <hm>
1470 ifw>4andw<12thenh(w+7)=h(w):s
=w+7                                <ga>
1480 ifw>7andw<15thenh(w-7)=h(w):u
=w-7                                <bl>
1490 ifw>14thenh(w-14)=h(w):h(w-7)
=h(w):v=w-14:o=w-7                  <gm>
1500 ns=ns+10:ifns>312thenns=an:z=
z+1                                 <mo>
1510 ifz=5thenz=1:oo=1:gosub120     <mh>
1520 lo=0:return                    <cm>
1530 rem =lied spielen=========== <nd>
1540 fori=1tonz                     <lg>
1550 wait65297,8:ifs(i)=0thenforp=
1tol(i):nextp:goto1570              <ef>
1560 sound1,s(i),l(i):sound2,s(i)+
2,l(i)                              <np>
1570 nexti                          <oo>
1580 return                         <pc>
1590 rem =aufloesungszeichen====== <ll>
1600 q=dec(q$):h=zp-2:xy=1          <fb>
1610 w=(zp-(z-1)*40-4)/2:a(w)=-h(w
):gosub2240                         <df>
1620 ifw<5thena(w+7)=-h(w):a(w+14)
=-h(w)                              <fe>
1630 ifw>4andw<12thena(w+7)=-h(w)   <nd>
1640 ifw>7andw<15thena(w-7)=-h(w)   <kp>
1650 ifw>14thena(w-7)=-h(w):a(w-14
)=-h(w)                             <pm>
1660 lo=0:return                    <gp>
1670 rem =taktstrich============= <fl>
1680 h=(z-1)*40+20:q=dec(q$)        <kd>
1690 gosub2240:xy=1                 <fe>
1700 fori=1to18                     <km>
1710 a(i)=0                         <fc>
1720 nexti                          <ei>
1730 ifq<>0thenh(q)=0:q=0           <ip>
1740 ifr<>0thenh(r)=0:r=0           <oh>
```

```
1750 ifs<>0thenh(s)=0:s=0                <cm>
1760 ift<>0thenh(t)=0:t=0                <dp>
1770 ifo<>0thenh(o)=0:o=0                <lm>
1780 ifu<>0thenh(u)=0:u=0                <km>
1790 ifv<>0thenh(v)=0:v=0                <ac>
1800 lo=0:return                         <ig>
1810 rem =notenzahl ueberschritten       <hp>
1820 graphic0,1                          <gj>
1830 printcu$cu$cu$tab(3)"mehr als
 500 toene gehen nicht."                 <mc>
1840 printtab(3)cu$"bitte melodie
speichern und neu"                       <ge>
1850 printtab(3)cu$"dimensionieren
 (zeile 2300)"                           <fp>
1860 printtab(3)cu$cu$cu$"taste dr
uecken"                                  <on>
1870 getkeyx$:graphic1,0:return          <ig>
1880 rem =note loeschen==========        <ek>
1890 iflo=1thenreturn                    <ob>
1900 ifxy>0andnz=0then1970               <of>
1910 ifxy>0andnz>0thenxy=0:goto193
0                                        <ho>
1920 nz=nz-1:ifnz<0thennz=0              <ch>
1930 ns=ns-10:ifns<anthenns=312:z=
z-1:ifz<1thenz=1:ns=an                   <dj>
1940 h=(z-1)*40+8                        <oe>
1950 gosub2180:gshapecn$,ns,pa-6         <jo>
1960 lo=1:return                         <md>
1970 h=(z-1)*40                          <io>
1980 ns=ns-10:an=an-10:ifns<12then
ns=12:an=12:xy=xy-1                      <hd>
1990 ifxy<0thenxy=0                      <fc>
2000 fori=htoh+120step40                 <mm>
2010 gshapecn$,ns,i                      <ge>
2020 nexti                               <pl>
2030 lo=1:return                         <da>
2040 rem =wiederholungszeichen====       <md>
2050 ifnz>500thengosub1820:return        <of>
2060 h=int(zp/42)                        <gj>
2070 h=40*h+20:q=val(q$)                 <dd>
2080 gosub2240:xy=1                      <pf>
2090 ifq$="0"thenreturn                  <ho>
2100 g=we-wa                             <ga>
2110 ifg+nz>500thengosub1820:retur
n                                        <bm>
2120 fori=we+1towe+g                     <ic>
2130 s(i)=s(i-g):l(i)=l(i-g)             <dd>
2140 nexti:nz=nz+g                       <ci>
2150 wa=0:we=0                           <en>
2160 return                              <ho>
2170 rem =pfeil positionieren=====       <jf>
2180 pa=(z-1)*40+6                       <ac>
2190 pe=z*40                             <bm>
2200 ifzp>pethenh=zp:zp=pe               <mj>
2210 ifzp<pathenh=zp:zp=pa               <bc>
2220 gshapecl$,0,h-4:gshapep$,0,zp       <fi>
2230 return                              <al>
2240 gshapen$(q),ns,h,2:ns=ns+10:i
fns>312thenns=an:z=z+1                   <of>
2250 ifz=5thenz=1:oo=1:gosub120          <ck>
2260 xy=0:return                         <ai>
2270 rem =pausen setzen==========        <kj>
2280 q=asc(q$)-55                        <dn>
2290 pp=2↑(16-(q-4))                     <ln>
2300 h=(z-1)*40-20*(pp=256)-21*(pp
=128)-20*(pp=64)-20*(pp=32)-20*(pp
=16)                                     <gb>
2310 gosub2240                           <kn>
2320 nz=nz+1:ifnz>500thengosub1820       <gf>
2330 s(nz)=0:l(nz)=3.5*pp+4*(q-8)+
20                                       <pa>
2340 lo=0:return                         <bo>
2350 rem =variablen==============        <ng>
2360 zp=6:nz=0:wa=0:we=0:ne=312:an
=12:ns=12:la=8                           <ae>
2370 dim s(500),l(500),n$(16),h(18
),a(18),t(18,2)                          <hm>
2380 cu$=chr$(17):co$=chr$(145)          <ko>
2390 fori=1to18                          <mg>
2400 forj=0to2                           <ng>
2410 readt(i,j)                          <aa>
2420 nextj,i                             <kk>
2430 char,1,1,"werden instruktione
n gewuenscht?(j/n)"                      <hm>
2440 getkeyx$                            <no>
2450 ifx$="n"thenreturn                  <bg>
2460 ifx$<>"j"then2430                   <bg>
2470 scnclr:char,1,1,"mit diesem p
rogramm koennen musik-"                  <kk>
2480 char,1,3,"stuecke von einem n
otenblatt abge-"                         <ef>
2490 char,1,5,"schrieben werden. m
it cursor up/down"                       <kf>
2500 char,1,7,"wird der pfeil (lin
ker rand) auf die"                       <cf>
2510 char,1,9,"entsprechende noten
linie gestellt."                         <dn>
2520 char,1,11,"mit '7' oder '8' w
erden die # oder b,"                     <id>
2530 char,1,13,"mit '1' - '6' die
notenwerte,"                             <kf>
2540 char,1,15,"mit '0' oder '9' d
ie wiederholungen"                       <in>
2550 char,1,17,"mit 'a' das aufloe
sungszeichen"                            <hh>
2560 char,1,19,"mit 'b' die taktst
riche und mit"                           <hb>
2570 char,1,21,"'c' - 'g' die paus
en eingegeben."                          <nm>
2580 char,10,24,"taste druecken"         <lg>
2590 getkeyyx$                           <bm>
2600 scnclr:char,1,1,"weitere tast
enbedeutungen:"                          <cm>
2610 char,1,3,"'l'= lied laden; 's
'= lied speichern"                       <ih>
2620 char,1,5,"'h'= programm beend
```

```
en"                                  <pc>
2630 char,1,7,"'n'= speicher loesc
hen, neuanfang"                      <hp>
2640 char,1,9,"'p'= lied spielen"    <bp>
2650 char,1,11,"'+' verlaengert di
e tondauer um die"                   <kg>
2660 char,5,13,"haelfte"             <jb>
2670 char,1,15,"'-' loescht die le
tzte eingabe"                        <kh>
2680 char,1,17,"der taktstrich ist
 stets zu setzen,"                   <ak>
2690 char,1,19,"wenn auch im musik
stueck ein solcher"                  <gg>
2700 char,1,21,"erscheint (senkrec
hter strich ueber"                   <kk>
2710 char,1,23,"5 notenlinien)"      <ij>
2720 char,10,24,"taste druecken"     <cc>
2730 getkeyyx$                       <ka>
2740 scnclr:char,1,1,"die noten, d
ie zwischen den wiederho-"           <af>
2750 char,1,3,"lungszeichen stehen
, werden doppelt"                    <hd>
2760 char,1,5,"gespeichert. wenn n
ur das wiederho-"                    <hb>
2770 char,1,7,"lungszeichen '9' ge
setzt ist, wird"                     <mc>
2780 char,1,9,"das stueck vom anfa
ng bis zu diesem"                    <hb>
2790 char,1,11,"wiederholungszeich
en doppelt ge-"                      <kj>
2800 char,1,13,"speichert."          <be>
2810 char,1,15,"zusaetzliche # ode
r b gelten nur im"                   <bj>
2820 char,1,17,"takt, das aufloesu
ngszeichen 'a'"                      <al>
2830 char,1,19,"ebenfalls. # oder
b, die zu beginn"                    <bk>
2840 char,1,21,"geschrieben wurden
,gelten fuer das"                    <hn>
2850 char,1,23,"ganze stueck."+b$+
" taste druecken"                    <ic>
2860 getkeyx$:return                 <mh>
2870 input#15,aa,bb$,cc,dd           <el>
2880 ifaa<>63then2950                <ag>
2890 ifaa=63thenprintcu$tab(3)na$"
 existiert"                          <dd>
2900 printcu$tab(3)"soll ueberschr
ieben werden? (j/n)                  <jn>
2910 getkeyx$:ifx$="n"thenu=1:clos
e15:close2:return                    <mn>
2920 ifx$<>"j"then2910               <ef>
2930 na$="@0:"+na$:close15:close2    <co>
2940 u=2:return                      <jl>
2950 ifaa=0thenreturn                <bd>
2960 printaa;bb$,cc,dd:close2:clos
e15:u=3                              <bb>
2970 print"taste druecken"           <lc>
2980 getkeyx$:return                 <on>
2990 rem =notendatas============= <ee>
3000 data911,917,923                 <jm>
3010 data904,911,917                 <kb>
3020 data889,897,904                 <jc>
3030 data872,881,889                 <mj>
3040 data854,864,872                 <fn>
3050 data844,854,864                 <fj>
3060 data822,834,844                 <dg>
3070 data798,810,822                 <ej>
3080 data784,798,810                 <ab>
3090 data754,770,784                 <ib>
3100 data721,739,754                 <mg>
3110 data685,704,721                 <jc>
3120 data664,685,704                 <bf>
3130 data620,643,664                 <mn>
3140 data571,596,620                 <eg>
3150 data544,571,596                 <fm>
3160 data585,516,544                 <ic>
3170 data418,453,485                 <oi>
3180 rem aenderungen fuer datasett
enbetrieb                            <li>
3190 rem streiche 1050 - 1070        <on>
3200 rem streiche 1200 - 1240,2870
 - 2980                              <bd>
3210 rem aendere: 1090 open2,1,0,n
a$                                   <pf>
3220 rem aendere: 1200 open2,1,1,n
a$                                   <pa>
3230 rem aendere: 1300 close2        <mk>
3240 rem nachspann=============== <pd>
3250 b$=chr$(32):b2$=b$+b$           <li>
3260 b$=b2$+b2$:b$=b$+b$+b2$         <ji>
3270 return                          <dd>
3280 rem ======================== <jp>
3290 rem 60671 bytes memory          <oe>
3300 rem 08852 bytes program         <eg>
3310 rem 00203 bytes variables       <ji>
3320 rem 05580 bytes arrays          <ka>
3330 rem 00659 bytes strings         <bl>
3340 rem 12288 bytes graphic         <ha>
3350 rem 33089 bytes free (0)        <eh>
3360 rem ======================== <di>
```

Musik

Fortsetzung von Seite 109

Wenn zur Speicherung keine Diskette vorhanden ist, kann mit Datasette mit folgenden Programmänderungen gearbeitet werden:
Streiche Zeilen 1050–1070
Ändere: 1090 open2,1,0,na$

Streiche Zeilen 1200–1240 und 2870–2980
Ändere: 1200 open2,1,1,na$
Ändere: 1300 close2

Grundsätzliches zur Eingabe: Notenlinien abzählen, Pfeil am linken Rand auf die entsprechende Notenlinie setzen und eine der Tasten 1 bis 6, 7, 8, A oder C bis G drücken.
Für die Eingabe 0, 9, B, „+“ oder „–“ ist die Stellung des Pfeiles gleichgültig, ebenso für F, H, L, S, P, N.

Lesen Sie bitte weiter auf Seite 114

```
10 rem datamaker musik==========p4 <mp>
20 rem (p) commodore welt team     <ho>
30 rem ============================ <ng>
40 rem (c) by reiner hickel        <hg>
50 rem                             <pd>
60 rem basic v3.5                  <od>
70 rem plus4 (c16/116 + 64 kb)     <cd>
80 rem floppy (kassette anpassbar) <nb>
90 rem ============================ <jg>
100 graphic1,1:graphic0            <oa>
110 scnclr:input"name des liedes";
a$                                 <ka>
120 open2,8,2,a$+",s,r"            <jf>
130 input#2,nz                     <nb>
140 x=int(nz/3)                    <kk>
150 ifnz-3*x=0thenh=0:goto170      <bo>
160 if(nz+1)/3=int((nz+1)/3)thenh=
1:elseh=2                          <hm>
170 dims(nz+h),l(nz+h)             <jb>
180 fori=1tonz                     <pj>
190 input#2,s(i),l(i)              <ai>
200 nexti                          <gp>
210 close2:ifh=0then250            <bm>
220 fori=nz+1tonz+h                <bf>
230 s(i)=0:l(i)=0                  <bm>
240 nexti                          <ap>
250 nz=nz+h                        <kg>
260 ad=4100                        <gf>
270 fori=1tonz                     <ek>
280 s1=int(s(i)/256):s2=s(i)-s1*25
6                                  <kf>
290 l1=int(l(i)/256):l2=l(i)-l1*25
6                                  <ed>
300 pokead,s1:pokead+1,s2:pokead+2
,l1:pokead+3,l2                    <lj>
310 ad=ad+4                        <dk>
320 nexti:ed=4100+nz*4-1:ad=4100   <km>
330 input"beginn der datazeilen";d
a                                  <df>
340 input"schrittweite";sw         <ep>
350 scnclr:print"370data";ad;",";e
d;",";da;",";sw:print"goto380"     <bm>
360 poke1319,19:poke1320,13:poke13
21,13:poke239,3                    <nl>
370 rem =reservierte zeile======== <he>
380 restore370:x$=""               <pc>
390 readad,ed,da,sw                <pn>
400 ifad-1=edthenscnclr:printda"da
ta0,0":poke1319,19:poke1320,13:pok
e239,2:end                         <bn>
410 forj=1to3                      <ik>
420 s=peek(ad)*256+peek(ad+1)      <mo>
430 l=peek(ad+2)*256+peek(ad+3)    <hn>
440 x$=x$+str$(s)+","+str$(l)      <mj>
450 ifj<3thenx$=x$+","             <mo>
460 ad=ad+4                        <hn>
470 nextj:nd=da+sw                 <la>
480 scnclr:print"370data"ad","ed",
"nd","sw:printstr$(da)"data"x$:pri
nt"goto380"                        <lm>
490 poke1319,19:poke1320,13:poke13
21,13:poke1322,13:poke239,4:end    <ai>
500 rem um die datas dann zu verwe
nden                               <hg>
510 rem ist am besten folgendes pr
ogramm                             <jb>
520 rem zu verwenden wegen der pau
sen:                               <nf>
530 rem 10 la=8: rem (lautstaerke) <km>
540 rem 20 restore:rem dataanfang
           lied                    <oi>
550 rem 30 volla:do:reads,l:getx$:
           ifx$<>""thenexit        <ii>
560 rem 40 ifs=0andl>0thenfori=1to
           l:nexti:goto70          <of>
570 rem 50 ifs=0andl=0then exit    <pb>
580 rem 60 sound1,s,l              <pj>
590 rem 70 wait65297,la:loop:retur
           n(hauptprogramm)        <ed>
600 rem la im waitbefehl entsprich
t der vol-einstellung              <ig>
610 rem die anweisung wartet, bis
der ton zu ende gespielt ist       <io>
620 rem zeile 370 ist die fuer das
 programm wichtige datazeile       <di>
630 rem aenderungen fuer datasette
nbetrieb                           <li>
640 rem aendere: 110 open2,1,0,na$ <pm>
650 rem ========================== <ge>
660 rem = achtung !!!            = <al>
670 rem = programm darf nicht    = <db>
680 rem = umnummeriert werden!   = <jg>
690 rem ========================== <cn>
700 rem  p r o g r a m m e n d e   <ao>
710 rem ========================== <ne>
```

Fortsetzung von Seite 113

2. Datamaker Musik

Dokumentation:

Zeilen 100–210	Laden des Liedes von D.
Zeilen 220–320	Schreiben der Noten mit Tondauer in den Speicher DEC 4100–6100
Zeilen 330–360	Eingabe der ersten Zeilennnummer der Data und Schrittweite. (Diese Daten werden in Zeile 370 als Arbeitsdatazeile übernommen.)
Zeilen 380–490	Umwandeln der Lieddaten in Datazeilen.
Zeilen 500–640	Anmerkungen mit kurzem Programm zum Abspielen der Melodie mit Ausstiegmöglichkeit.

Pausen erscheinen in den Data mit Notenwert 0 und Tondauer größer als 96.

Nachdem die Datas geschrieben sind, DELETE –710 eingeben, dann stehen nur noch die Datazeilen im Speicher.

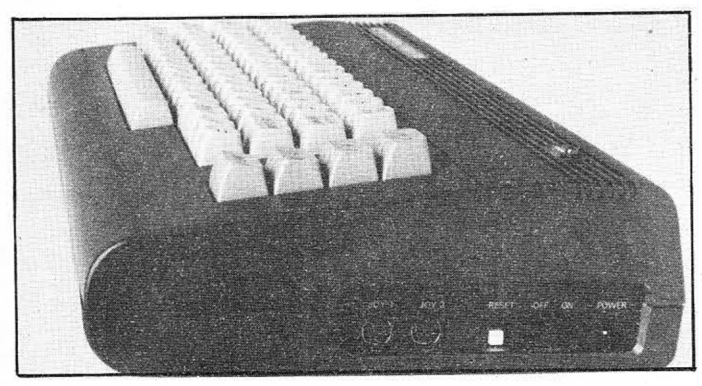
Bitte beachten, daß der Grafikmodus eingeschaltet ist und der BASIC-Anfang bei Hex $4000 liegt. □

STARTMENÜ

Programmanwahl per Tastendruck

Mühseliges Eingeben des Ladebefehls und des Programmnamens gehören der Vergangenheit an. Mit dem Startmenü als erstem File auf der Diskette geht es viel einfacher.

Das Startmenü sollten Sie sich auf jede frisch formatierte Diskette kopieren. Steht dieses Programm an erster Stelle, so braucht es nur den Druck der Commodore- und der Run-Taste. Das Inhaltsverzeichnis ist nun in Menüform zu sehen. Nur die PRG-Files werden aufgelistet. Ein revers dargestellter Buchstabe signalisiert, mit welchem Tastendruck ein bestimmtes Programm zu laden ist.



Sind mehr Programme auf der Diskette, als auf den Bildschirm passen, so ist dies für das Startmenü auch kein Handicap. Mit der Return- oder der Cursor-down-Taste werden die nächsten 16 Directory-Einträge sichtbar. Mit Pfeil-Taste oder Cursor-up läßt sich bequem wieder zurückblättern. Ist ein bestimmtes Programm ausgewählt, wird es automatisch geladen und zugleich mit RUN gestartet. Für Maschinenprogramme, die eventuell anders aufzurufen sind, läßt sich dieses Startprogramm allerdings nicht verwenden.

LESEN DES INHALTSVERZEICHNISSES

In Zeile 210 des Programmes STARTMENUE.BAS steht die erforderliche OPEN-Anweisung. Wer allerdings denkt, mit INPUT#-Befehlen würden die einzelnen Einträge gelesen werden können, irrt sich. Erst am Ende des Directories erscheint der für den INPUT so wichtige Return-Code mit dem ASCII-Wert 13. Der Eingabepuffer wäre unterdessen schon längst übergelaufen, was das System mit einem String-too-long-Error quittieren würde. Daher ist in BASIC der GET#-Befehl zu verwenden.

LESEN DES FILE-NAMENS

Ob die einzelnen Einträge mit einer Null abgeschlossen werden oder nicht, darum brauchen wir uns nicht zu kümmern. Wir wollen ja nur den File-Namen lesen. Dieser ist eingeschlossen in Anführungszeichen, die den ASCII-Wert 34 besitzen. Die Unterroutine in den Zeilen 110 bis 150 liest die nach dem ersten Anführungszeichen stehenden Buchstaben des File-Namens in die Variable A$ ein. Bei Erreichen des zweiten Anführungszeichens wird die Routine beendet und der File-Name liegt zur weiteren Bearbeitung bereit.
Ist das Directory bereits zuende gelesen, so sorgt Zeile 120 für den vorzeitigen Rücksprung. Für den ersten Directory-Eintrag, der den Header enthält, haben wir keine Verwendung. Erst die nachfolgenden Einträge sind für uns interessant. In eine DO-LOOP-Schleife lesen wir alle File-Namen ein, bis das Return-Zeichen uns das Ende signalisiert.

LESEN DES FILE-TYPS

Da nur die Programm-Files auf der Diskette für uns wichtig sind, führen wir eine Selektion durch. Dazu ist der File-Typ zu lesen. Dieser folgt auf den File-Namen, durch mehrere Leerzeichen getrennt. Sein Ende ist ebenso durch Leerzeichen markiert. Mit einer zweiten Unterroutine in den Zeilen 160 bis 190 bekommen wir ihn in den Griff. Wir lesen weiter bis zum ersten Zeichen, das kein Leerzeichen ist. Die folgenden Buchstaben werden erfaßt, bis das anschließende Leerzeichen wieder den Rücksprung auslöst.

MIT MASCHINENCODE GEHT ES SCHNELLER

GET-Befehle, mit denen jeder Buchstabe einzeln in einer Schleife gelesen werden muß, sind ziemlich langsam. Den Wunsch nach Schnelligkeit kannn nur ein Maschinenprogramm erfüllen, das die GET-Routinen ersetzt. Das Programm STARTMENUE.MC liest das Directory in etwa derselben Zeit, die auch der Directory-Befehl brauchen würde. Mit der Routine BASIN – es geht auch mit GETIN – werden die gewünschten Zeichen in den Eingabepuffer gebracht. Mit zwei POKEs setzen wir das GET-Flag und signalisieren, daß die Eingabe nicht von der Tastatur erfolgt ist. Mit einem SYS-Aufruf erreichen wir dann die Übergabe der im Eingabepuffer liegenden Zeichen an unsere Stringvariable.

BASIC-FASSUNG ANPASSBAR

Die Fassung STARTMENUE.BAS ist besonders für diejenigen interessant, die das Programm gerne für den C128 oder den C64 umschreiben möchten. Beim C64 müssen alle DO-LOOPs durch GOTO-Sprünge oder FOR-NEXT-Schleifen ersetzt werden. Die Methode mit der Funktionstasten-Belegung hat für einen C64 natürlich keinen Sinn. Wer nicht das ROM des C128 nach einem SYS-Befehl zur Simulation eines Funktionstasten-Druckes durchforsten will, tut gut daran, die althergebrachte Methode zur Übergabe von Befehlen in den Direktmodus zu verwenden.
Vor Programmende ist der Bildschirm zu löschen und die Lade- sowie Startanweisung an die richtigen Bildschirmpositionen zu schreiben. Danach sind der Home-Code und die erforderliche Anzahl von Return-Codes in den Tastaturpuffer zu POKEn und die Anzahl der gePOKEten Zeichen dem Tastaturindex zu übergeben. Nach Beendigung des Programmes übergibt der Rechner durch die im Tastaturpuffer gespeicherten Return-Codes – die auf den Bildschirm geschriebenen Anweisungen an sich selbst.
Die Funktionstasten-Belegung beim C16 hat den Vorteil, daß nicht zu berücksichtigen ist, in welcher Bildschirmzeile nach einer Ladeanweisung der Cursor wohl stehen mag. Der Funktionstasten-Puffer ist schließlich groß genug, daß er nicht nur ein paar Return. Zeichen, sondern die vollständigen Anweisungen aufnehmen kann. □

STARTMENUE.MC

```
10 rem startmenue.mc===========c16  <co>
20 rem (p) commodore welt team      <ho>
30 rem ==========================   <ng>
40 rem (c) by roland engelhardt     <nf>
50 rem (v)    a. mittelmeyer    mc  <gd>
60 rem                              <ah>
70 rem basic v3.5                   <nl>
80 rem c16/116/plus4 + disk         <pi>
90 rem ==========================   <jg>
100 fori=818to875                   <mn>
110 reada:pokei,a:next              <ne>
120 data 032,145,167,201,013,208    <dh>
130 data 001,096,201,034,208,244    <em>
140 data 162,000,032,145,167,201    <eb>
150 data 034,240,011,157,000,002    <el>
160 data 232,224,089,144,241,076    <jd>
170 data 076,204,076,049,144,169    <nh>
180 data 034,141,068,003,208,214    <lg>
190 data 162,000,169,032,141,068    <mf>
200 data 003,032,145,167,201,032    <gk>
210 data 240,249,208,219            <gb>
220 dimv$(207):dimy$(15)            <nk>
230 open1,8,0,"$0":poke2035,1:sys6
5478                                <oi>
240 poke19,1:poke17,64:sys853       <da>
250 do:sys853:ifst<>0thenexit       <gp>
260 sys37210b$:sys860:sys37210a$    <jc>
270 ifa$="prg"thenv$(x)=b$:x=x+1    <oi>
280 loop:sys65484:close1            <ph>
290 re$=chr$(28):rn$=chr$(18)       <ln>
300 fl$=chr$(130):fo$=chr$(132)     <km>
310 bk$=chr$(144):rf$=chr$(146)     <al>
320 b$=chr$(32)                     <lf>
330 b2$=b$+b$:b4$=b2$+b2$           <ja>
340 b5$=b4$+b$:b$=b5$+b5$           <nm>
350 forx=0to15:y$(x)=chr$(65+x):ne
xtx                                 <bk>
360 scnclr:print                    <pi>
370 printtab(10)rn$re$b$b$rf$bk$    <af>
380 printtab(10)rn$re$b5$"start-me
nue"b4$                             <cf>
390 printtab(10)rn$re$b$b$rf$bk$    <na>
400 print                           <lm>
410 forx=0to15step2:printrn$re$" "
y$(x)" "rf$bk$" "v$(x+y);           <go>
420 printtab(20)rn$re$" "y$(x+1)"
"rf$bk$" "v$(x+y+1):print:nextx     <fb>
430 printtab(12)rn$re$fl$b$b5$rf$b
k$fo$                               <ca>
440 printtab(12)rn$re$fl$b2$"waehl
e file"b2$rf$fo$bk$                 <lh>
450 printtab(12)rn$re$fl$b$b5$rf$b
k$fo$                               <ai>
460 getkeya$:ify>170then480         <nn>
470 ifa$=chr$(13)ora$=chr$(17)then
y=y+16:goto360                      <cp>
480 ify<16then500                   <md>
490 ifa$="^"ora$=chr$(145)theny=y-
16:goto360                          <ff>
500 ifa$<"a"ora$>"p"then460         <ei>
510 a=asc(a$)-65                    <po>
520 scnclr                          <pm>
530 key4,"dL"+chr$(34)+v$(a+y)+chr
$(13)+"rU"+chr$(13)                 <ng>
540 poke2035,3:sys56364             <og>
550 rem ==========================  <pp>
560 rem  p r o g r a m m e n d e    <hp>
570 rem ==========================  <mo>
```

STARTMENUE.BAS

```
10 rem startmenue.bas==========c16  <pk>
20 rem (p) commodore welt team      <ho>
30 rem ==========================   <ng>
40 rem (c) by roland engelhardt     <nf>
50 rem                              <pd>
60 rem                              <ah>
70 rem basic v3.5                   <nl>
80 rem c16/116/plus4 + disk         <pi>
90 rem ==========================   <jg>
100 goto200                         <od>
110 b$=""                           <co>
120 get#1,c$:ifc$=chr$(13)thenretu
rn                                  <pf>
130 ifc$<>chr$(34)then120           <ma>
140 get#1,c$:ifc$<>chr$(34)thenb$=
b$+c$:goto140                       <go>
150 return                          <ll>
160 a$=""                           <hg>
170 get#1,c$:ifc$=chr$(32)then170   <hm>
180 ifc$<>chr$(32)thena$=a$+c$:get
#1,c$:goto180                       <fe>
190 return                          <al>
200 dimv$(207):dimy$(15)            <jg>
210 open1,8,0,"$0"                  <gl>
220 gosub110                        <ij>
230 do:gosub110:ifst<>0thenexit     <mi>
240 gosub160                        <na>
250 ifa$="prg"thenv$(x)=b$:x=x+1    <pl>
260 loop:close1                     <ao>
270 re$=chr$(28):rn$=chr$(18)       <hg>
280 fl$=chr$(130):fo$=chr$(132)     <ad>
290 bk$=chr$(144):rf$=chr$(146)     <me>
300 b$=chr$(32)                     <jb>
310 b2$=b$+b$:b4$=b2$+b2$           <an>
320 b5$=b4$+b$:b$=b5$+b5$           <fb>
330 forx=0to15:y$(x)=chr$(65+x):ne
xtx                                 <kf>
340 scnclr:print                    <fh>
350 printtab(10)rn$re$b$b$rf$bk$    <ie>
360 printtab(10)rn$re$b5$"start-me
nue"b4$                             <ei>
370 printtab(10)rn$re$b$b$rf$bk$    <af>
380 print                           <jd>
```

```
390 forx=0to15step2:printrn$re$" "
y$(x)" "rf$bk$" "v$(x+y);              <be>
400 printtab(20)rn$re$" "y$(x+1)"
"rf$bk$" "v$(x+y+1):print:nextx        <pe>
410 printtab(12)rn$re$fl$b$b5$rf$b
k$fo$                                  <ib>
420 printtab(12)rn$re$fl$b2$"waehl
e file"b2$rf$fo$bk$                    <lo>
430 printtab(12)rn$re$fl$b$b5$rf$b
k$fo$                                  <ca>
440 getkeya$:ify>170then460            <nl>
450 ifa$=chr$(13)ora$=chr$(17)then
y=y+16:goto340                         <cd>
460 ify<16then480                      <nh>
470 ifa$="^"ora$=chr$(145)theny=y-
16:goto340                             <eh>
480 ifa$<"a"ora$>"p"then440            <po>
490 a=asc(a$)-65                       <mp>
500 scnclr                             <nd>
510 key4,"dL"+chr$(34)+v$(a+y)+chr
$(13)+"rU"+chr$(13)                    <bp>
520 poke2035,3:sys56364                <gl>
530 rem =========================      <bm>
540 rem  p r o g r a m m e n d e       <po>
550 rem =========================      <pp>
```

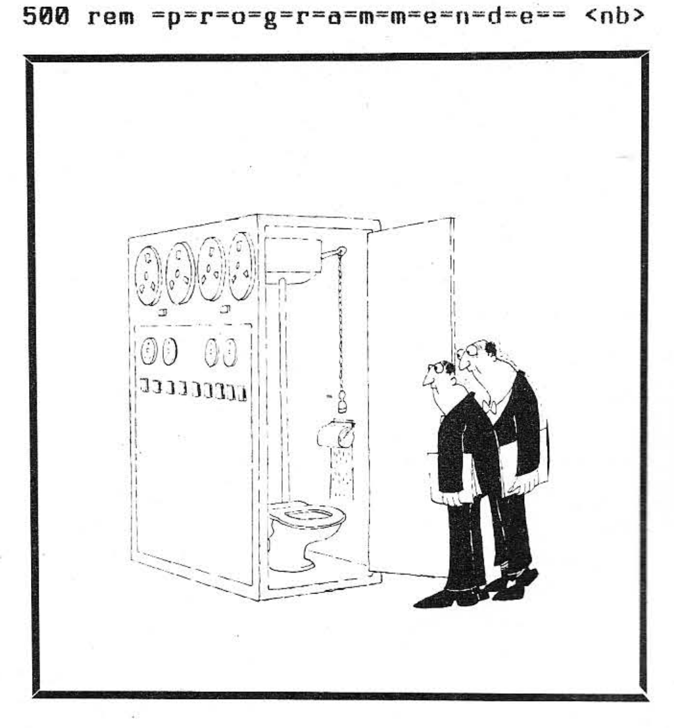
FARBDEMO

Alle Farben auf dem Monitor

Nicht nur als Demo, welche Farben ein C16 hat, dient dieses Programm. Auch der Monitor kann optimal eingestellt werden. Alle Farben und Helligkeitsstufen werden auf dem Monitor dargestellt. Sie können durch Drehen an Farb-, Helligkeits- und Kontrast-Regler die beste Farbwiedergabe erreichen. □

FARBDEMO

```
10 rem farbdemo===============c16      <lj>
20 rem (p) commodore welt team         <ho>
30 rem ==========================      <ng>
40 rem (c) by detlef lokay             <hg>
50 rem                                 <pd>
60 rem                                 <ah>
70 rem basic v3.5                      <nl>
80 rem c16/116/plus4                   <ki>
90 rem ==========================      <jg>
100 graphic1,1:clr:a=1:color1,a,0      <nc>
110 scnclr:box1,50,20,250,100          <kc>
120 do                                 <nl>
130 color1,a,5:color4,a,5              <om>
140 char1, 7,5,"**** c-16 farbdemo
 ****"                                 <mg>
150 char1,7,9,"mit space starten"      <hn>
160 char1,7,10,"mit space beenden"     <nh>
170 a=a+1:forx=1to100:next             <fl>
180 ifa=16thena=1                      <nj>
190 geta$:ifa$<>""then210              <hp>
200 loop                               <hn>
210 graphic0:color1,1                  <ap>
220 printcl$                           <fk>
230 d= 5:e=25                          <ab>
240 graphic1,1                         <em>
250 color1,1                           <dd>
260 forx=1to8                          <mi>
270 a=17:b=5:c=20                      <nj>
280 do                                 <bn>
290 box1,b,d,24,e                      <ip>
300 b=b+24-5                           <li>
310 box1,c,d,c,e                       <hm>
320 a=a-1: c=c+20-1                    <im>
330 loopuntila=0                       <ka>
340 d=d+24:e=e+24                      <nd>
350 nextx:e=0                          <af>
360 l=16:k=7:b=15:e=15                 <kb>
370 fory=1to16                         <la>
380 k=7                                <kp>
390 forz=1to8                          <il>
400 color1,l,k                         <ik>
410 paint1,b,e                         <nj>
420 k=k-1:e=e+24                       <bh>
430 nextz                              <bn>
440 rem k=7                            <oo>
450 l=l-1:e=15                         <hm>
460 b=b+24-5                           <bi>
470 nexty                              <ll>
480 getx$:ifx$=""then480               <mc>
490 graphic0,1:color1,1:color4,7,5
:end                                   <pi>
500 rem =p=r=o=g=r=a=m=m=e=n=d=e==     <nb>
```



GORGUL

Gorgul, ein Tyrann übelster Sorte, hat sich der Herrschaft in Deinem bis dahin glücklichen Heimatland bemächtigt. Aufgrund Deines großen Mutes wurdest du von der Bevölkerung deines Landes gebeten sie von Gorgul zu befreien. Du erfüllst ihre Bitte und brichst in den Palast Gorguls ein, um ihn zu finden und zu töten. Das ist die Lage zu Beginn des Adventures. Jetzt mußt Du versuchen, Gorgul in seinem Palast zu finden, dabei seinen Wächtern auszuweichen oder sie zu bekämpfen, von den Gegenständen, die Dir begegnen, zu bewältigen.

Um Deine Aktionen zu steuern, mußt du Anweisungen mit ein oder zwei Wörtern eingeben.

Folgende Befehle sind Ein-Wort-Kommandos:

– Hilfe	Dieser Befehl sollte in jedem Raum zuerst eingegeben werden. Man erhält dann einen Hinweis, der meistens nützlich, manchmal aber auch nutzlos ist.
– Kämpfe	
– Iss	
– Gepäckliste	Listet die Gegenstände auf, die man trägt.

Folgende Befehle sind Zwei-Wort-Kommandos, d.h. es muß zusammen mit dem Befehl noch ein Wort zur näheren Bestimmung eingegeben werden (z.B. "Nimm Messer" oder "Gehe Nord"):

– Benutze	
– Gehe	mit diesem Befehl kann man sich von einem Raum in den nächsten bewegen.
– Werfe	
– Zerbreche	
– Nimm	mit diesem Befehl kann man einen Gegenstand, den man in einem Raum sieht, mitnehmen. – Gegenteil: Weglege
– Klettere	
– Verschiebe	
– Weglege	
– Betrachte	dieser Befehl gibt einem nähere Informationen über einen Gegenstand

Programmaufbau:

Zeile 100 – 340	1. Hauptschleife
Zeile 350 – 1030	Unterprogramme der 1. Hauptschleife
Zeile 1040 – 1250	2. Hauptschleife
Zeile 1260 – 1690	Unterprogramme der 2. Hauptschleife
Zeile 1720 – 1860	Daten für Ortsbeschreibungen und Gegenstände

Programmtechnische Besonderheiten:
– Ortsbeschreibungen sind in DATA-Zeilen abgelegt
– es gibt 2 Hauptschleifen. Beim Auftreten irgendeiner Gefahr für den Spieler (Wächter, Feuer, Gorgul) wird von der 1. Hauptschleife in die 2. Hauptschleife gesprungen. In der 2. Hauptschleife hat der Spieler weniger Befehle zur Verfügung. Ein falscher Befehl bedeutet den sofortigen Tod.

Anmerkung:
Ich habe mich bemüht das Spiel so zu schreiben, daß man beim Abtippen so wenig wie möglich von der Handlung des Spieles erfährt. Dies ging allerdings etwas zu Lasten der Übersichtlichkeit des Programmes und der Sprache des Spiels. □

```
10 rem gorgul==================c16  <gm>
20 rem (p) commodore welt team      <ho>
30 rem ==========================   <ng>
40 rem (c) by wolfgang markert      <pb>
50 rem         oberwerrn            <mm>
60 rem                              <ah>
70 rem basic v3.5                   <nl>
80 rem c16/116/plus4                <ki>
90 rem ==========================   <jg>
100 clr:gosub1990:poke1344,64:gosu
b1700:kr=1                          <fp>
110 ifkr=1thenkr$="ausreichend":el
seifkr=0thenkr$="nicht mehr vorhan
den"                                <fd>
120 scnclr:print"du befindest dich
:":printl$(l)                       <di>
130 print:print"du kannst gehen na
ch: ";:ifd%(l,0)<>0thenprint"nord
";                                  <bp>
140 ifd%(l,1)<>0thenprint"ost ";    <cn>
150 ifd%(l,2)<>0thenprint"sued ";   <gg>
160 ifd%(l,3)<>0thenprint"west"     <am>
170 print:print"kraft: ";tab(14)kr
$                                   <pm>
180 print:print"du siehst:";:c=0    <ea>
190 fork=1to7:ifo%(k)=lthenprintta
b(14)ol$(k):c=1                     <ka>
200 nextk                           <hb>
210 if(w1=0andl=2)or(w2=0andl=5)or
(w3=0andl=11)or(w4=0andl=12)thenc=
1:goto1040                          <jh>
220 if(lw=0andl=16)or(h=0andl=17)o
r(f=0andl=18)orl=19thenc=1:goto104
0                                   <od>
230 ifc=0thenprinttab(14)"nichts i
nteressantes"                       <de>
240 forx=1to40:print"=";:nextx      <gh>
250 i$="":j$="":m$=""               <dk>
260 input"was jetzt";i$             <jl>
270 x=len(i$):fory=1tox:k$=mid$(i$
,y,1):ifk$=" "thengoto280:elsenext
y                                   <fl>
280 y=y-1:j$=left$(i$,3):y=y+2:m$=
mid$(i$,y,3)                        <hk>
290 ifj$="geh"thengoto350:elseifj$
="nim"thengoto400:elseifj$="weg"th
engoto460                           <hl>
300 ifj$="gep"thengoto510:elseifj$
="bet"thengoto540:elseifj$="iss"th
engoto630                           <ap>
310 ifj$="hil"thengoto650:elseifj$
="wer"thengoto810:elseifj$="zer"th
engoto880                           <ne>
320 ifj$="kle"thengoto920:elseifj$
="ver"thengoto970:elseifj$="kae"th
engoto980                           <mb>
330 ifj$="ben"thengoto1000          <mb>
340 print"entschuldige bitte,aber
ich verstehe"b4$"dich nicht!":goto
250                                 <fp>
350 ifm$="nor"andd%(l,0)<>0thenl=d
%(l,0):goto110                      <en>
360 ifm$="ost"andd%(l,1)<>0thenl=d
%(l,1):goto110                      <jm>
370 ifm$="sue"andd%(l,2)<>0thenl=d
%(l,2):goto110                      <ae>
380 ifm$="wes"andd%(l,3)<>0thenl=d
%(l,3):goto110                      <ee>
390 print"dahin kann ich nicht geh
en!":goto250                        <jg>
400 fork=1to7:ifo$(k)=m$ando%(k)=-
1thenprint"du hast den gegenstand
doch schon":goto250                 <cn>
410 nextk                           <lo>
420 fork=1to7:ifo%(k)=lando$(k)=m$
thengoto440                         <df>
430 nextk:print"ich kann den gegen
stand hier nicht sehen":goto250     <ga>
440 ifgl=5thenprint"du hast schon
5 gegenstaende.du kannst  nichts m
ehr nehmen":goto250                 <bb>
450 gl=gl+1:o%(k)=-1:print"ok-du h
ast den gegenstand":forx=1to2000:n
extx:goto110                        <gf>
460 fork=1to7:ifm$=o$(k)thengoto48
0                                   <ga>
470 nextk:print"diesen gegenstand
gibt es in diesem"b5$"spiel gar ni
cht":goto250                        <nh>
480 fork=1to7:ifm$=o$(k)ando%(k)=-
1thengoto500                        <id>
490 nextk:print"du hast diesen geg
enstand doch gar nicht":goto250     <ad>
500 o%(k)=l:gl=gl-1:print"du hast
den gegenstand abgelegt":forx=1to2
000:nextx:goto110                   <gb>
510 b=0:print"du traegst:";:forx=1
to7:ifo%(x)=-1thenprinttab(14)ol$(
x):b=1                              <df>
520 nextx:ifb=0thenprinttab(14)"ni
chts"                               <eh>
530 goto250                         <pa>
540 if(l=1oro%(4)=-1)andm$="tas"th
enprint"sie ist blau":goto250       <dh>
550 if(l=3oro%(3)=-1)andm$="rev"th
enprint"er ist geladen und schussb
ereit":goto250                      <pe>
560 if(l=5oro%(2)=-1)andm$="feu"th
enprint"er ist rot":goto250         <aj>
570 if(l=7orl=14oro%(1)=-1)andm$="
spi"thengoto580:elsegoto590         <jg>
580 print"man kann ihn leicht zerb
rechen":goto250                     <al>
590 if(l=9oro%(5)=-1)andm$="sei"th
enprint"es sieht morsch aus":goto2
50                                  <ld>
```

```
600 if(l=10oro%(6)=-1)andm$="wur"t
henprint"sie sieht appetitlich aus
":goto250                          <eg>
610 if(l=11oro%(7)=-1)andm$="sch"t
henprint"es ist sehr scharf":goto2
50                                 <ca>
620 print"ich kann den gegenstand
hier nicht sehen":goto250          <mf>
630 ifl<>10thenprint"du kannst nur
 in der kueche etwas essen!":goto2
50                                 <jc>
640 print"gut! jetzt hast du wiede
r neue kraefte":kr=1:forx=1to2000:
nextx:goto110                      <ck>
650 ifl=1thenprint"morgenstund hat
 gold im mund":goto250             <ki>
660 ifl=3thenprint"reden ist silbe
r-schweigen ist gold":goto250      <ge>
670 ifl=4thenprint"wenn man keine
kraft mehr hat,muss man  etwas ess
en":goto250                        <dl>
680 ifl=7thenprint"scherben bringe
n glueck-oder auch nicht":goto250  <hn>
690 ifl=9thenprint"fuer eine versc
hlossene tuer braucht maneinen sch
luessel":goto250                   <le>
700 ifl=10thenprint"alles hat ein
ende nur die wurst hat"b4$"zwei":g
oto250                             <hn>
710 ifl=13thenprint"der hund ist d
er beste freund eines"b5$"mannes":
goto250                            <ph>
720 ifl=14thenprint"mancher ist sc
hon,als er sich im spiegelsah,ersc
hrocken":goto250                   <de>
730 ifl=15thenprint"glueck und gla
s,wie leicht bricht das":goto250   <dl>
740 printc4$b3$"du hast folgende b
efehle:"                           <hk>
750 printc4$b3$"gehe,nimm,weglege,
betrachte,"                        <ao>
760 printb3$"iss(f3-taste),gepaeck
liste(f1-taste),"                  <pb>
770 printb3$"hilfe(help-taste),kae
mpfe(f2-taste),"                   <ed>
780 printb3$"benutze,werfe,zerbrec
he,klettere und"                   <de>
790 printb3$"verschiebe."          <mb>
800 goto250                        <ob>
810 ifl=12andm$="sei"ando%(5)=-1th
en850                              <ii>
820 fork=1to7:ifm$=o$(k)thengoto84
0                                  <ji>
830 nextk:print"diesen gegenstand
gibt es in diesem"b5$"spiel gar ni
cht":goto250                       <je>
840 print"nichts geschieht":goto25
0                                  <bm>
850 print"gut gemacht!"            <dd>
860 print"mit dem befehl 'klettere
 hinunter'"b6$"kannst du dich jetz
t auf den unteren"                 <nh>
870 print"balkon herunterseilen":k
m=1:goto250                        <ja>
880 ifl=14andm$="spi"thengoto900   <do>
890 print"das kann ich leider nich
t zerbrechen":goto250              <aj>
900 print"hinter einem der spiegel
 war eine ver-  borgene tuer,die n
ach osten geht"                    <fp>
910 print"diese tuer ist jetzt geo
effnet":d%(14,1)=17:forx=1to3000:n
extx:goto110                       <fe>
920 ifkm=1andl=12andm$="hin"thengo
to940                              <fo>
930 print"dahin kann ich nicht kle
ttern":goto250                     <hd>
940 print"du kletterst an dem seil
 herunter,doch  kurz bevor du unte
n ankommst reisst"                 <bh>
950 print"das seil und du faellst
auf den unteren balkon.du kannst n
icht mehr hoch."                   <fc>
960 forx=1to3000:nextx:o%(5)=0:gl=
gl-1:l=13:goto110                  <dm>
970 print"nichts geschieht":goto25
0                                  <pp>
980 print"warum willst du hier kae
mpfen? ich kann hier keinen gegner
 sehen"                            <ad>
990 goto250                        <mf>
1000 fork=1to7:ifm$=o$(k)then1020  <ml>
1010 nextk:print"diesen gegenstand
 gibt es in diesem"b5$"spiel gar n
icht!":goto250                     <el>
1020 print"es hat wenig sinn jetzt
 ";ol$(k);" zu benutzen":goto250   <lc>
1030 end                           <je>
1040 if(w1=0andl=2)or(w2=0andl=5)o
r(w3=0andl=11)thenprinttab(14)"ein
en waechter"                       <dn>
1050 if(w4=0andl=12)thenprinttab(1
4)"einen waechter"                 <mf>
1060 iflw=0andl=16thenprinttab(14)
"den leibwaechter gorguls"         <mj>
1070 ifh=0andl=17thenprinttab(14)"
einen grossen hund,der",,left$(qr$
,4)+"dich anknurrt"                <bp>
1080 iff<>0orl<>18then1110         <gp>
1090 printtab(14)"sobald du den ra
um be-",,left$(qr$,4)+"trittst fae
ngt dieser"                        <dn>
1100 printtab(14)"zu brennen an."  <cg>
1110 ifl=19thenprinttab(14)"gorgul
"                                  <dp>
1120 forx=1to40:print"=";:nextx    <kf>
```

```
1130 i$="":j$="":m$=""                  <gp>
1140 input"was jetzt";i$                <nj>
1150 x=len(i$):fory=1tox:k$=mid$(i
$,y,1):ifk$=" "thengoto1160:elsene
xty                                     <lb>
1160 y=y-1:j$=left$(i$,3):y=y+2:m$
=mid$(i$,y,3)                           <in>
1170 ifj$="kae"then1200:elseifj$="
ben"then1260:elseifj$="hil"then135
0                                       <nn>
1180 ifj$="bet"then1400:elseifj$="
gep"then1460                            <ka>
1190 goto1490                           <og>
1200 ifl=16orl=17orl=18orl=19then1
490                                     <eh>
1210 if(l=2orl=5orl=11orl=12)andkr
=0then1490                              <km>
1220 printc4$"der waechter greift
dich an, aber als"b3$"er sieht,das
s du staerker bist,"                    <oe>
1230 print"fluechtet er"                <ef>
1240 kr=0:ifl=2thenw1=1:elseifl=5t
henw2=1:elseifl=11thenw3=1:elseifl
=12thenw4=1                             <ig>
1250 forx=1to4000:nextx:goto110         <dg>
1260 ifl=2orl=5orl=11orl=12orl=16t
hen1490                                 <kj>
1270 ifl=17andm$="wur"ando%(6)=-1t
hen1300                                 <le>
1280 ifl=18andm$="feu"ando%(2)=-1t
hen1320                                 <hp>
1290 ifl=19andm$="spi"ando%(1)=-1t
hen1340:elsegoto1490                    <ko>
1300 print"der hund schnappt sie d
ir aus der hand  und verschwindet
mit ihr"                                <nl>
1310 forx=1to4000:nextx:o%(6)=0:h=
1:goto110                               <pj>
1320 print"du kannst das feuer loe
schen"                                  <fj>
1330 forx=1to4000:nextx:f=1:goto11
0                                       <og>
1340 print"gorgul sieht sich im sp
iegel an und":gosub1900                 <hm>
1350 ifl=2orl=5orl=11orl=12thenpri
nt"versuche zu kaempfen":goto1130       <np>
1360 ifl=16thenprint"du bist in de
r naehe gorguls":goto1130               <gm>
1370 ifl=17thenprint"schau dir den
 hund an und denke nach!":goto1130      <em>
1380 ifl=18thenprint"das feuer ist
 sehr warm":goto1130                    <ip>
1390 ifl=19thenprint"schau dir gor
gul an und denke nach!":goto1130        <hj>
1400 if(l=2orl=5orl=11orl=12)andm$
="wae"thengoto1410:elsegoto1420         <ga>
1410 print"er traegt eine rosarote
 uniform":goto1130                      <be>
1420 ifl=16andm$="lei"thenprint"er
 soll unbesiegbor sein":goto1130        <ib>
1430 ifl=17andm$="hun"thenprint"er
 ist hungrig":goto1130                  <km>
1440 ifl=19andm$="gor"thenprint"er
 ist sehr haesslich":goto1130           <ka>
1450 goto1490                           <id>
1460 b=0:print"du traegst:";:forx=
1to7:ifo%(x)=-1thenprinttab(14)ol$
(x):b=1                                 <ce>
1470 nextx:ifb=0thenprinttab(14)"n
ichts"                                  <ao>
1480 goto1130                           <cc>
1490 printcl$c4$left$(qr$,4)cl$"du
 hast das falsche gemacht!!":print      <hb>
1500 if(l=2orl=5orl=11orl=12)andkr
=0thengoto1540                          <bh>
1510 if(l=2orl=5orl=11orl=12)andkr
=1thengoto1570                          <ic>
1520 ifl=16thengoto1600:elseifl=17
thengoto1620                            <fk>
1530 ifl=18thengoto1630:elseifl=19
thengoto1640                            <aj>
1540 printc$"der waechter hat dich
 angegriffen und"                       <aa>
1550 print"konnte dich,da du keine
 kraft mehr"                            <dm>
1560 print"hattest,leicht ueberwae
ltigen":goto1650                        <id>
1570 printc$"der waechter hat dich
 angegriffen und"                       <fh>
1580 print"konnte dich,da du falsc
h gehandelt"                            <af>
1590 print"hattest,leicht ueberwae
ltigen":goto1650                        <gm>
1600 printc$"der leibwaechter gorg
uls konnte dich"b4$"mit seiner ueb
erlegenen kampfkunst                    <ne>
1610 print"leicht besiegen":goto16
50                                      <db>
1620 printc4$"der hund stuerzte si
ch auf dich und zer-fleischte dich
":goto1650                              <ho>
1630 printc4$"das feuer schloss di
ch ein und du ver-  branntest":got
o1650                                   <ne>
1640 printc4$"gorgul stuerzte sich
 auf dich und"b4$b3$"toetete dich"
:goto1650                               <gd>
1650 printc$c$"damit waere deine m
ission gescheitert"                     <on>
1660 printc$"jetzt besteht keine h
offnung"                                <ig>
1670 printc4$"mehr das land von go
rgul zu befreien"                       <gi>
1680 printc4$c4$c4$"willst du noch
 einmal spielen(j/n)";:inputa$          <ko>
1690 ifa$="j"thenrun:elseend            <nc>
```

```
1700 color1,1:color0,14,5:printcl$
"bitte warten":diml$(19):dimd%(19,
3)                                    <gn>
1710 fork=1to19:readl$(k),d%(k,0),
d%(k,1),d%(k,2),d%(k,3):next:l=6      <hl>
1720 dataim bad eines maennergemac
hs,,,,2,im schlafzimmer eines maen
nergemachs,3,1,,                      <ag>
1730 dataim salon eines maennergem
achs,,4,2,,in einem vorzimmer,8,5,
,3                                    <gb>
1740 dataim salon eines frauengema
chs,,,6,4,im schlafzimmer eines fr
auengemachs,5,,,7                     <an>
1750 dataim bad eines frauengemach
s,,6,,                                <hb>
1760 datain einem geschmackvoll ei
ngerichtetem   esszimmer,11,9,4,12    <af>
1770 datain einem langen flur;im o
sten ist eine  verschlossene tuer,
10,,,8                                <ng>
1780 datain einer kueche,,,9,,in e
iner gutbestueckten bibliothek,,,8
,                                     <pa>
1790 dataauf einem balkon;ein stoc
kwerk unter dirkannst du wieder ei
nen balkon sehen                      <cn>
1800 data,8,,,auf einem balkon,,14
,,                                    <dh>
1810 datain einem praechtigen saal
;dessen waende mit spiegeln verkle
idet sind,,,15,13                     <ld>
1820 datain einem flur,14,,,16,in
einer grossen empfangshalle,,15,,     <na>
1830 datain einem kleinen salon,,,
18,14,in einem langen gang,17,,19,    <bn>
1840 dataim arbeitszimmer gorguls,
18,,,                                 <od>
1850 dimo$(7):dimol$(7):dimo%(7):f
ork=1to7:reado$(k),ol$(k),o%(k):ne
xtk                                   <ah>
1860 dataspi,einen spiegel,7,feu,e
inen feuerloescher,5,rev,einen rev
olver,3                               <bh>
1870 datatas,eine taschenlampe,1,s
ei,ein seil,9,wur,eine wurst,10,sc
h,ein schwert,11                      <gf>
1880 key1,"gepaeckliste"+chr$(13):
key2,"kaempfe"+chr$(13):key3,"iss"
+chr$(13)                             <ni>
1890 key8,"hilfe"+chr$(13):return     <eb>
1900 print"faellt vor schreck tot
um":forx=1to2000:nextx                <ag>
1910 scnclr:printc4$c4$left$(qr$,1
5)rn$"gratuliere!"rf$                 <je>
1920 printc4$c4$c4$c3$"du hast dei
ne gefaehrlich mission"               <ce>
1930 printc4$c3$"erfolgreich beend
et.jetzt wird endlich"                <bp>
1940 printc4$c3$"wieder frieden in
 das land einkehren."                 <oj>
1950 printc4$c4$c4$c3$"willst du n
och einmal spielen(j/n)";:inputa$     <fl>
1960 ifa$="j"thenrun:elseend          <ad>
1970 rem nachspann ===============    <hc>
1980 rem * farbcodes/steuercodes *    <gf>
1990 c4$=chr$(017):rn$=chr$(018):c
$=c4$                                 <ee>
2000 c3$=chr$(029):rf$=chr$(146)      <hj>
2010 cl$=chr$(147)                    <ap>
2020 rem ********* zeichenfolgen *    <pm>
2030 for q=1 to 40                    <pa>
2040 qr$=qr$+c3$                      <ah>
2050 next q:b$-chr$(32):b2$-b$+b$     <jm>
2060 b3$=b2$+b$:b4$=b3$+b$            <jf>
2070 b5$=b4$+b$:b6$=b5$+b$:return     <cm>
2080 rem ======================       <gb>
2090 rem 12277 bytes memory           <pn>
2100 rem 09262 bytes program          <ok>
2110 rem 00070 bytes variables        <kk>
2120 rem 00321 bytes arrays           <fb>
2130 rem 01133 bytes strings          <hh>
2140 rem 01491 bytes fre(0)           <ll>
2150 rem ======================       <fc>
```

HIRES-GRAFIK MIT NUR 16 KBYTE

3-D-Design-Golf

Dem C16 verbleiben im HIRES-Grafikmodus nur noch etwa zwei KByte Programmspeicher. Fast unmöglich mag es scheinen, damit noch viel anzufangen. Daß es dennoch geht, beweist dieses sehr interessante Spiel, das wirklich das Letzte aus dem Rechner herausholt.

Viele C16-User, die noch keine Speichererweiterung haben, waren bestimmt schon verärgert, daß bei Nutzung der High-Res.-Grafik zehn KByte vom RAM abgezogen werden, so daß nur zwei KByte zum Programmieren übrigbleiben. Da es unter diesen Umständen kaum Sinn hat, in High-Res. zu programmieren, sind vielen C16-Besitzern die Wirkungsweisen der Grafikbefehle unbekannt.
Mit dem Programm 3-D-DESIGN-GOLF möchte ich drei Ziele verfolgen:
- Demonstration der High-Res.-Grafik;
- maximale Speicherplatz-Ausnutzung;
- auf Dauer interessantes Spielgeschehen.

Herausgekommen ist ein 3-D-Golfspiel, bei dem es möglich ist, Golfbahnen nach dem eigenen Geschmack zu entwerfen und abzuspeichern, um sie dann in perspektivischer Darstellung zu bespielen. Trotz des ungemein knappen Speicherplatzes konnten Raffinessen wie Windbeeinflussung und Screenwechsel beim „Putten“ berücksichtigt werden.
Für Tippfaule dürfte somit 3-D-DESIGN-GOLF ein kleiner Leckerbissen sein – geringe Mühe wird mit einem beeindrucken-

den Resultat belohnt. 3-D-DESIGN-GOLF besteht aus zwei Teilen. Der GOLF-DESIGNER dient dazu, Golfbahnen zu konstruieren, die später vom Spielprogramm GOLF-PLAYER eingeladen werden. Vom GOLF-PLAYER existieren zwei Versionen. Wer nur 16 KByte zur

SPIELANLEITUNG

Verfügung hat, muß das Programm GOLF-PLAYER.C16 verwenden. Da der Programmspeicher bis zur Grenze in Anspruch genommen wird, ließ sich keine Abfrage für Diskette mehr unterbringen.
Daher muß der C16/116-User, auch wenn er eine Floppy besitzt, dennoch mit Datasette arbeiten. Die Programme selber können dabei auf Diskette bleiben, nur die mit dem GOLF-DESIGNER entworfenen Golfbahnen sind auf Kassette abzuspeichern. Da der GOLF-DESIGNER auf das Speichermedium zugreift, von dem er eingeladen wurde, muß derjenige, der ihn auf Diskette hat, die Bahnen aber auf Kassette speichert, eine kleine Änderung vornehmen: In Zeile 360 ist die Variable G mit G=1 auf den Wert Eins zu setzen. Der OPEN-Befehl in Zeile 380 kann dann die Datasette ansprechen.
Wer mehr als 16 KByte zur Verfügung hat, sollte die Version GOLF-PLAYER.P4 verwenden, da er ansonsten entweder seinen Rechner vorher mit der Hand immer auf 16 KByte zurücksetzen oder durch Ein- und Ausschalten der Grafik die Variablenzeiger richtigstellen müßte. Ebenso wie der GOLF-DESIGNER greift der GOLF-PLAYER.P4 auf das Gerät zu, von dem es geladen wurde. Ist dieses Gerät die Datasette, so wird das nächste sequentielle Kassetten-File geladen.
Bei der Diskettenstation wird der Name des zu ladenden Spielfeldes abgefragt. Wer eine Floppy hat, den GOLF-PLAYER.P4 von Diskette einlädt und dennoch die Golfbahnen von der Datasette laden will, kann dies durch eine kleine Programmänderung verwirklichen. Er braucht in Zeile 120 nur G=PEEK(174) in G=1 zu verwandeln. Der Datasettenbenutzer speichert zweckmäßigerweise zuerst den GOLF-DESIGNER und dann den GOLF-PLAYER ab. Der Zählerstand danach sollte aufgeschrieben werden, denn es gilt, später dort men auszuschließen. Die Abspeicherungen der Bahnen erfolgen zweckmäßigerweise hinter dem Spielprogramm. Spulen Sie dazu das Band bis zum Ende des GOLF-PLAYER vorwärts. Nach dem Speichern können neue Bahnen konstruiert werden, bis zur Beendigung die „E"-Taste gedrückt wird.
Um 3-D-DESIGN-GOLF zu spielen, muß der GOLF-PLAYER geladen werden. Steuerungen erfolgen mit dem Joystick (Port eins). Das Programm startet damit, daß die Runden-(Bahnen-)Anzahl erfragt wird, wonach Sie zum La-



die Golfbahnen abzuspeichern.
Laden Sie dazu den Konstruktionsteil GOLF-DESIGNER und konstruieren Sie eine Golfbahn nach Ihren Vorstellungen mit dem Joystick in Port eins.
Joystickbewegungen ohne Feuerknopf-Betätigung hinterlassen ein Rasenfeld, mit Feuerknopf folgt ein Wasserfeld. Demnach braucht bei Korrekturen nur das Gegenteil vorgenommen zu werden. Beachten Sie bei der Konstruktion, daß von links nach rechts gespielt wird; die Start- und Zielfelder ermittelt der Computer per Zufall.
Zur Abspeicherung einer fertigen Bahn drücken Sie „S". Die Bezeichnung dient dazu, Verwechslungen mit anderen Program-

den der ersten Bahn aufgefordert werden. Anschließend baut sich die Spielbahn aus der Perspektive des Spielers auf, wobei der Abschlagpunkt das unterste mittlere Feld ist.
Zur Bestimmung der Schlagstärke dienen zwei Anzeigen: Die obere Anzeige stellt die nach vorn gerichtete Schlagstärke dar. Joystick nach vorn = Zunahme; Joystick nach hinten = Abnahme. Die untere Anzeige dient der Darstellung der seitwärts gerichteten Schlagstärke. Joystick nach rechts oder links bedeutet, daß die entsprechende Richtung verstärkt wird.
Der waagerechte Strich auf der unteren Anzeige erlaubt Rückschlüsse über Windrichtung und -Stärke. Strich weit rechts: starker Wind von links.
Der Wind verlängert oder verkürzt die seitliche Kraft und wirkt sich verkürzend auf die vorwärts gerichtete Kraft aus.
Der Abschlag erfolgt über den Feuerknopf. Danach verschwindet der unten abgebildete Ball und taucht nach Berechnung der Flugbahn auf dem berechneten Feld wieder auf. Anschließend wird das Bild von dieser neuen Position aus wieder aufgebaut. Landete der Ball im Wasser, erfolgt der Bildschirmaufbau an der alten Position.

EINPUTTEN

Das Zielfeld ist durch eine weiße Stange gekennzeichnet. Nachdem es erreicht ist, wird der Ball in das Loch geputtet. Hierzu wird das Zielfeld formatfüllend in Dunkelgrün dargestellt, wobei das Zielloch als weiße Ellipse zu sehen ist. Um den Ball in das Loch zu schlagen, wird der Joystick nach vorn gedrückt und bei genügender Schlagkraft (aus der Anzeige ersichtlich) losgelassen; eine seitliche Krafteinstellung ist nicht nötig. Die neue Entfernung zum Loch ergibt sich aus der Differenz eines zu kurzen oder zu langen Schusses und der Distanz zum Zielloch. Wurde der Ball erfolgreich eingeputtet, erscheint eine Anzeige der Schlagzahl innerhalb der Runde, der Summe der bisherigen Schläge und der Rundenzahl.
Nun kann die nächste Bahn geladen werden.

PROGRAMM-BESCHREIBUNG

Zum Programm GOLF-DESIGNER:

- Die Zeilen 110 bis 160 dienen zum Bildschirmaufbau.
- Innerhalb der Zeilen 170 bis 240 erfolgt 8-mal eine Joystickabfra-

ge; jeweils mit Kontrolle auf Randüberschreitung, Cursordarstellung innerhalb der entsprechenden Unterprogramme (280 bis 330) und nachfolgender Einberechnung in F$() (=späteres Spielfeld).

- Der POKE-Befehl in Zeile 250 löscht den Tastaturpuffer, so daß die Tasten „S" und „E" abgefragt werden können (Zeile 250 und 260).
- Die Taste „S" leitet zur SAVE-Routine (Zeile 340 bis 380) über, wobei F$() abgespeichert und zum Programmanfang zurückgesprungen wird.

Zum Programm GOLF-PLAYER.C16:

- Nach INPUT der gewünschten Rundenzahl (Zeile 1) beginnt der Ladevorgang.
- In Zeile 5 werden die Start- und Zielpunkte per Zufall ermittelt.
- Das nachfolgende Hauptprogramm befindet sich innerhalb der Zeilen 6 bis 22. Es wird verlassen, wenn in Zeile 6 festgestellt wird, daß die aktuelle Ballposition dem Zielpunkt entspricht.

Zeile 7 stellt sicher, daß sich der Ball auf einem Rasenfeld befindet.
Nachfolgend wir innerhalb der Zeilen 8 bis 10 ein Gitter aufgezeichnet, wobei die Hintergrundfarbe (0, 7, 2) mit der Gitterfarbe (2, 7, 2) identisch ist.
Zeile 11 kürzt den Bildschirmrand, um nicht berechnete Felder zu löschen.
Zeile 12 füllt die Gitterfelder mit Hilfe des PAINT-Befehls bezogen auf Farbzone 1 aus, um Überschneidungen mit Zone 0 und 2 zu vermeiden.
Diese Arbeit bewerkstelligt das Unterprogramm 31 unter Verwendung der in Zeile 3 und 4 abgelegten DATA. Das Ausfüllen ist davon abhängig, ob der Wert X=6 (Rasenfläche) vorhanden ist. Der Wert „X" errechnet sich durch den VAL-Befehl aus F$().

Zur Bedeutung der DATAs:

A = Anfangsposition des Pixel-Cursors
B = Seitliche Schrittweite des Cursors
C = Vertikale Position des Cursors
K = Differenz zwischen J und L (Variablenliste)

Die Zeilen 15 bis 22 ermöglichen den Abschlag. Hierbei können mit dem Joystick die Werte „R" (seitliche Richtung– und „V" (Vorwärts) verändert werden, was mittels der Unterprogramme 25 und 27 dargestellt wird.
Nach dem Abschlag (Zeile 21) folgt die Berechnung der neuen Position in Zeile 23. Anschließend läßt sich mit Hilfe der Variablen „TR" die Routine in Zeile 12 zur Darstellung des Balls an neuer Position nochmals verwenden. Jetzt findet ein Rücksprung zur Zeile 6 statt, worauf ein Bildschirmaufbau an der neuen Stelle erfolgt.
Wurde im Unterprogramm 28 (abgerufen in Zeile 14) festgestellt, daß diese das Spielfeld überschreitet oder daß ein Wasserfeld vorliegt, wird die alte Position wieder übernommen.
Falls das Zielfeld erreicht wurde (Zeile 6), findet innerhalb des Programmabschnitts 38 bis 45 ein Screenwechsel zum „Putten" statt.
Die Entfernung Ball/Loch wird durch Zufall in Zeile 38 ermittelt und das Loch anschließend in Form einer Ellipse dargestellt, deren Abmessungen und Position innerhalb der Zeilen 39 bis 41 errechnet wurden.
Die Joystickabfrage (42 bis 43) bezieht sich allein auf die Schlagkraft, wobei der Wert „V" erhöht und mit Hilfe des Unterprogramms 27 sichtbar gemacht wird.
Bei Loslassen des Joysticks wird die neue Ballposition in Bezug auf den Lochabstand berechnet (Zeile 44) und bei fehlender Übereinstimmung erneut dargestellt.
Andernfalls wird der Bildschirm gelöscht. Bevor erneut geladen wird (Zeile 45), werden die aktuellen Werte angezeigt.
Da es wegen des geringen Speicherplatzes nicht möglich war, Leerstellen und schwer identifizierbare Zeichen in gewohnter Weise durch Stringvariablen zu ersetzen, sind ein paar zusätzliche Bemerkungen angebracht:
In Zeile 11 haben wir zweimal je drei Leerzeichen, in Zeile 24 zweimal je zwei Leerzeichen vor uns. Um gar 20 Leerzeichen handelt es sich in Zeile 25. Wenn Sie das Zeichen in Zeile 27, das aussieht, wie ein kleines Dächlein, nicht auf der Tastatur finden, so liegt das an unserem Drucker. Geben Sie die Pfeiltaste ein, die Sie auf der Tastatur durch Shift und 0 erreichen. Die Anzahl der Leerzeichen nach dem Doppelpunkt beläuft sich auf elf.
Das Programm GOLF-PLAYER.P4 unterscheidet sich nur geringfügig von der 16-KByte-Version. Es ist mit Programmkopf und Nachspann versehen und umnumeriert. In Zeile 100 werden die Variablenzeiger richtiggestellt und die Routine zur Initialisierung bestimmter Variablen für Steuercodes aufgerufen. Die Zeilen 120 bis 150 ersetzen die einzeilige Laderoutine der C16-Version. Bei Nichtvorfinden des gewählten Spielfeldes wird beim Diskettenbetrieb wiederholt der Spielfeldname abgefragt. Die Schlußzeile der C16-Version wurde durch die Zeilen 590 bis 620 ersetzt, da es angehmer sein dürfte, lediglich eine einzige Taste zu drücken, als den BASIC-Befehl CONT zum Fortfahren im Programm einzugeben.

VARIABLENLISTE

Zu Listing 1:

A	= Universelle FOR-NEXT-Variable
F$()	= Spielfeld
K$	= Tastaturabfrage
N$	= Namensgebung für Golfbahnen
PH	= Horizontale Cursor-Position
PV	= Vertikale Cursor-Position

Zu Listing 2:

A,B,C,D,E	= Universelle FOR-NEXT-Variablen
F$()	= Spielfeld
GP	= Gesamtpunktzahl
J	= Horizontale Ballposition
K	= Differenz zwischen J und L
L	= Anfangsposition bei Spielfeldabtastung
M	= Vertikale Ballposition
N	= Vertikale Abtastposition beim Bildschirmaufbau
O	= Umrechnungswert für J
P	= Umrechnungswert für M
R	= Seitenkraft
RG	= Gesamtrunden
RU	= Rundenzahl
TP	= Teilpunkte
TR	= Umschalter für Feld- u. Balldarstellung
V	= Vorwärts-Kraft
W	= Wind
W1	= Rechenwert zur Windberechnung
X	= VAL-Abgreif-Wert von F$()
Z	= Zielpunkt

```
10 rem golf-designer===========c16 <ap>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by peter bergen         <pk>
50 rem                             <pd>
60 rem                             <ah>
70 rem basic v3.5                  <nl>
80 rem c16/116/plus4               <ki>
90 rem ========================== <jg>
100 gosub 390                      <bj>
110 color0,2,6:color1,6,4:scnclr:f
ora=1to9:f$(a)="666666666666666666
66":next                           <np>
120 fora=10to18:char1,10,a,rn$+b$+
rf$:next                           <ig>
130 char1,7,2,bk$+"joystick (1) oh
ne fire =":color1,6,4:char1,32,2,r
n$+" "+rf$                         <if>
140 char1,7,4,bk$+"joystick (1)"+b
2$+"mit fire =":color1,7,2:char1,3
2,4,rn$+" "+rf$                    <bl>
150 char1,7,6,bk$+"s = save / e =
ende"                              <fl>
160 ph=1:pv=1:gosub300             <om>
170 ifjoy(1)=3andph<20thengosub310
:mid$(f$(pv),ph,1)="6":ph=ph+1:gos
ub300                              <fc>
180 ifjoy(1)=7andph>1thengosub310:
mid$(f$(pv),ph,1)="6":ph=ph-1:gosu
b300                               <fd>
190 ifjoy(1)=131andph<20thengosub2
90:mid$(f$(pv),ph,1)="7":ph=ph+1:g
osub280                            <mm>
200 ifjoy(1)=135andph>1thengosub29
0:mid$(f$(pv),ph,1)="7":ph=ph-1:go
sub280                             <pg>
210 ifjoy(1)=5andpv<9thengosub310:
mid$(f$(pv),ph,1)="6":pv=pv+1:gosu
b300                               <ij>
220 ifjoy(1)=1andpv>1thengosub310:
mid$(f$(pv),ph,1)="6":pv=pv-1:gosu
b300                               <ej>
230 ifjoy(1)=133andpv<9thengosub29
0:mid$(f$(pv),ph,1)="7":pv=pv+1:go
sub280                             <hf>
240 ifjoy(1)=129andpv>1thengosub29
0:mid$(f$(pv),ph,1)="7":pv=pv-1:go
sub280                             <ja>
250 poke239,0:getkeyk$:ifk$="s"the
n340                               <gn>
260 ifk$="e"thenscnclr:color1,1:st
op                                 <kl>
270 goto170                        <ka>
280 color1,7,2:goto320             <jm>
290 color1,7,2:goto330             <hl>
300 color1,6,4:goto320             <gj>
310 color1,6,4:goto330             <jl>
320 char1,ph+9,pv+9,rn$+"*"+rf$:re
turn                               <bc>
330 char1,ph+9,pv+9,rn$+" "+rf$:re
turn                               <ea>
340 char1,0,22,bk$:print"spielfeld
bezeichnung (max.12 zeichen)"b2$;:
inputn$                            <ji>
350 iflen(n$)>12then340            <kp>
360 g=peek(174)                    <ol>
370 ifg=8thenn$=n$+",s,w"          <mf>
380 open1,g,g,n$:fora=1to9:print#1
,f$(a):next:close1:goto110         <ag>
390 rem nachspann ================ <oc>
400 rem * farbcodes/steuercodes  * <hl>
410 rn$=chr$(018):bk$=chr$(144)    <fl>
420 rf$=chr$(146):b$=chr$(32)      <he>
430 b2$=b$+b$:b$=b2$:fori=1to9     <ee>
440 b$=b$+b2$:next:return          <lg>
450 rem ========================== <do>
460 rem 12277 bytes memory         <kb>
470 rem 01710 bytes program        <mn>
480 rem 00063 bytes variables      <gi>
490 rem 00040 bytes arrays         <gc>
500 rem 00551 bytes strings        <ec>
510 rem 09913 bytes free (0)       <ha>
520 rem ========================== <cm>
```

GOLF-PLAYER.P4

```
10 rem golf-player=============p4 <fc>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by peter bergen         <pk>
50 rem                             <pd>
60 rem                             <ah>
70 rem basic v3.5                  <nl>
80 rem plus4 (c16/116 + 64 kb)     <fd>
90 rem ========================== <jg>
100 graphic1,1:clr:graphic0:gosub6
50                                 <cl>
110 scnclr:gp=0:vol5:a=rnd(1)-ti:i
nput"runden";rg                    <og>
120 g=peek(174):ifg=1thenopen1,1,0
:input#1,f$:goto150                <fj>
130 input"spielfeldname";na$       <aj>
140 open1,8,8,na$+",s,r":f$="":inp
ut#1,f$:iff$=""thenclose1:goto130  <fj>
150 f$(1)=f$:fora=2to9:input#1,f$(
a):next:close1:color4,2,6          <gh>
160 data20,30,143,2,10,23,115,3,8,
18,91,4,10,14,69,5,7,10.5,54,7,5,8
.4,41,9                            <kh>
170 data18,6.2,31,10,28,4.75,24,11 <hn>
180 tp=0:scnclr:m=1:j=int(rnd(1)*9
)+1:z=int(rnd(1)*9)+1              <eo>
190 ifj=zandm=20then510            <hc>
```

```
200 mid$(f$(j),1,1)="6":mid$(f$(z)
,20,1)="6":tr=1                      <ii>
210 color0,7,2:color1,6,4:color2,7
,2:color3,2,6:graphic3,1             <nd>
220 fora=-340to500step40:draw2,80,
0toa,200:next                        <ec>
230 b=40:c=235:d=5:fora=1to11:c=c-
b:draw2,0,cto159,c:d=d-.23:b=b-d:n
ext:tr=1                             <hl>
240 fora=0to24:char3,0,a,b3$,1:cha
r3,37,a,b3$,1:next                   <pc>
250 restore:ford=0to7:reada,b,c,k:
n=m+d:l=j-k:fore=ato160stepb         <ld>
260 iftr=1thengosub440:elsegosub48
0                                    <ao>
270 l=l+1:next:next:iftr=2thengosu
b410:goto190                         <ik>
280 r=0:v=0:w=int(rnd(1)*15)-7:gos
ub380:gosub400                       <ae>
290 b=40:e=80:c=145:gosub500:tr=2    <jg>
300 ifjoy(1)=3andr<80thenr=r+2:gos
ub380                                <ic>
310 ifjoy(1)=7andr>-80thenr=r-2:go
sub380                               <og>
320 ifjoy(1)=1andv<80thenv=v+2:gos
ub400                                <bh>
330 ifjoy(1)=5andv>0thenv=v-2:gosu
b400                                 <dd>
340 ifjoy(1)=128then360              <gd>
350 goto300                          <jh>
360 r=r/8:v=v/10:w1=w/12*abs(r):o=
j+int(r+w1+w/20*v+.5):p=m+int(v-(v
*(abs(w)/40))+.5)                    <pe>
370 tr=2:char1,19,17,b2$,1:char1,1
9,18,b2$,1:tp=tp+1:goto250           <io>
380 char1,4,23,"<:"+b$+b$+":>",1:d
raw2,r/2+64,185tor/2+64,190          <dl>
390 draw2,w*4+62,188tow*4+66,188:r
eturn                                <in>
400 char1,4,21,chr$(94)+": "+b$,1:
draw2,v/2+24,169tov/2+24,174:retur
n                                    <eb>
410 ifo<1oro>9orp>20thenreturn       <ci>
420 ifmid$(f$(o),p,1)="7"thenretur
n                                    <dl>
430 j=o:m=p:return                   <dh>
440 ifl<1orl>9orn>20then470:elsex=
val(mid$(f$(l),n,1))                 <oi>
450 ifx=6thenpaint1,e,c,1            <dm>
460 ifl=zandn=20thendraw3,e,ctoe,c
-b                                   <eo>
470 return                           <dn>
480 ifo=landp=nthengosub500          <hp>
490 return                           <gf>
500 circle3,e,c,int(b/15+.5):paint
3,e,c:return                         <ok>
510 color0,6,2:m=int(rnd(1)*8)+1     <in>
520 scnclr:v=0:gosub400:b=40:c=195
:d=6:fora=1to8:c=c-b                 <mb>
530 ifa=mthencircle3,90,c,b/2,b/4:
paint3,90,c                          <pd>
540 d=d-.23:b=int(b-d+.5):next:b=6
0:e=90:c=190:gosub500                <he>
550 ifjoy(1)<>1then550               <kg>
560 ifjoy(1)=1andv<80thenv=v+1:gos
ub400:goto560                        <dd>
570 tp=tp+1:v=int(v/8+.5):ifv=mthe
n580:elsex=m-v:m=abs(x):goto520      <fn>
580 graphic0,1:gp=gp+tp:ru=ru+1:pr
int,c4$c4$c4$;"tp:";tp,"gp:";gp,"r
u:";ru                               <ca>
590 poke239,0:print:ifru<>rgthen12
0                                    <jg>
600 print"weiter mit leertaste"      <ed>
610 geta$:ifa$<>" "then610           <nd>
620 rg=rg+1:goto180                  <mg>
630 rem nachspann================    <hl>
640 rem * farb- und steuercodes  *   <ai>
650 c4$=chr$(17):b$=chr$(32)         <cf>
660 b2$=b$+b$:b3$=b2$+b$             <nd>
670 b$=b3$+b3$+b2$+b2$:return        <kd>
680 rem ========================     <cc>
690 rem 60671 bytes memory           <bi>
700 rem 02360 bytes program          <fb>
710 rem 00196 bytes variables        <nm>
720 rem 00040 bytes arrays           <ha>
730 rem 00567 bytes strings          <dg>
740 rem 45220 bytes free (0)         <eb>
750 rem 12288 bytes graphic          <kd>
760 rem ========================     <jf>
```

GOLF-PLAYER.C16

```
1 scnclr:gp=0:vol5:a=rnd(1)-ti:inp
ut"runden";rg:g$=chr$(17)            <dj>
2 open1,1,0:fora=1to9:input#1,f$(a
):next:close1:color4,2,6             <ob>
3 data20,30,143,2,10,23,115,3,8,18
,91,4,10,14,69,5,7,10.5,54,7,5,8.4
,41,9                                <jd>
4 data18,6.2,31,10,28,4.75,24,11     <ig>
5 tp=0:scnclr:m=1:j=int(rnd(1)*9)+
1:z=int(rnd(1)*9)+1                  <ao>
6 ifj=zandm=20then38                 <jm>
7 mid$(f$(j),1,1)="6":mid$(f$(z),2
0,1)="6":tr=1                        <eb>
8 color0,7,2:color1,6,4:color2,7,2
:color3,2,6:graphic3,1               <op>
9 fora=-340to500step40:draw2,80,0t
oa,200:next                          <hj>
10 b=40:c=235:d=5:fora=1to11:c=c-b
:draw2,0,cto159,c:d=d-.23:b=b-d:ne
xt:tr=1                              <gm>
11 fora=0to24:char3,0,a,"    ",1:ch
ar3,37,a,"    ",1:next               <ij>
12 restore:ford=0to7:reada,b,c,k:n
=m+d:l=j-k:fore=ato160stepb          <mn>
```

GOLF-PLAYER.C16

```
13 iftr=1thengosub31:elsegosub35        <lb>
14 l=l+1:next:next:iftr=2thengosub
28:goto6                                <gn>
15 r=0:v=0:w=int(rnd(1)*15)-7:gosu
b25:gosub27                             <gg>
16 b=40:e=80:c=145:gosub37:tr=2         <nh>
17 ifjoy(1)=3andr<80thenr=r+2:gosu
b25                                     <ck>
18 ifjoy(1)=7andr>-80thenr=r-2:gos
ub25                                    <oi>
19 ifjoy(1)=1andv<80thenv=v+2:gosu
b27                                     <cj>
20 ifjoy(1)=5andv>0thenv=v-2:gosub
27                                      <ak>
21 ifjoy(1)=128then23                   <ki>
22 goto17                               <ha>
23 r=r/8:v=v/10:w1=w/12*abs(r):o=j
+int(r+w1+w/20*v+.5):p=m+int(v-(v*
(abs(w)/40))+.5)                        <ej>
24 tr=2:char1,19,17,"  ",1:char1,1
9,18,"  ",1:tp=tp+1:goto12              <jj>
25 char1,4,23,"<:
   :>",1:draw2,r/2+64,185tor/2+64,
190                                     <km>
26 draw2,w*4+62,188tow*4+66,188:re
turn                                    <dj>
27 char1,4,21,"^:              ",1:dr
aw2,v/2+24,169tov/2+24,174:return       <kc>
28 ifo<1oro>9orp>20thenreturn           <hi>
29 ifmid$(f$(o),p,1)="7"thenreturn      <ni>
30 j=o:m=p:return                       <kb>
31 ifl<1orl>9orn>20then34:elsex=va
l(mid$(f$(l),n,1))                      <kl>
32 ifx=6thenpaint1,e,c,1                <no>
33 ifl=zandn=20thendraw3,e,ctoe,c-
b                                       <hf>
34 return                               <nc>
35 ifo=landp=nthengosub37               <bo>
36 return                               <ng>
37 circle3,e,c,int(b/15+.5):paint3
,e,c:return                             <hp>
38 color0,6,2:m=int(rnd(1)*8)+1         <po>
39 scnclr:v=0:gosub27:b=40:c=195:d
=6:fora=1to8:c=c-b                      <ja>
40 ifa=mthencircle3,90,c,b/2,b/4:p
aint3,90,c                              <cb>
41 d=d-.23:b=int(b-d+.5):next:b=60
:e=90:c=190:gosub37                     <nc>
42 ifjoy(1)<>1then42                    <gg>
43 ifjoy(1)=1andv<80thenv=v+1:gosu
b27:goto43                              <aa>
44 tp=tp+1:v=int(v/8+.5):ifv=mthen
45:elsex=m-v:m=abs(x):goto39            <ee>
45 graphic0,1:gp=gp+tp:ru=ru+1:pri
nt,g$g$g$;"tp:";tp,"gp:";gp,"ru:";
ru                                      <ka>
46 poke239,0:ifru=rgthenstop:rg=rg
+1:goto5:else2                          <hd>
```

KOPIERPROGRAMM

Von Diskette auf Kassette? Kein Problem!

Wer Programme erstellt hat und diese auf Kassette vertreiben will, steht vor dem Problem, wie er die Kassetten kopieren soll. Das HiFi-Kassettendeck mit zweiten Laufwerk ist nicht die wahre Lösung.

Signale vom Kassettenrekorder unterscheiden sich etwas von den direkt aus dem Computer kommenden Signalen. Dies führt zu Qualitätseinbußen bei den kopierten Kassetten. Bei etwas dejustiertem Tonkopf sind sie bereits nicht mehr zu lesen. Spezielle Kopiergeräte für Computerkassetten sind nahezu unerschwinglich. Die beste Methode ist es, die Programme von Diskette in den Computer einzulesen und dann auf die Kassette abzuspeichern.
Mit der Hand, jedes Programm einzeln, ist dies freilich viel zu mühsam.

PRG- UND SEQ-FILES

Das Programm DISK-TO-TAPE greift auf ein sequentielles File zu, das die Namen der zu kopierenden Files enthält. Es können Programme und sequentielle Files kopiert werden. Beim normalen Hereinladen und Speichern von Maschinenprogrammen mit LOAD "FILENAME",8,1 und SAVE"FILENAME" wird der ganze Bereich vom BASIC-Anfang bis zum Programmende gespeichert. Ein kurzes Maschinenprogramm, das den Bereich von $3E00 bis $3FFF in Anspruch nimmt und nur 512 Byte lang ist, würde auf der Kassette plötzlich zwölf KByte in Anspruch nehmen.

DISK-TO-TAPE verstellt in so einem Fall die SAVE-Anfangsadresse und sorgt damit für die geringste Länge. Beim Einladen weist der Lade-Zeiger auf eine Adresse in den System-Pages, ab welcher drei Nullen abgelegt sind. Ein Erzeugen von Linkadressen im Maschinenprogramm, in der irrigen Annahme, es handle sich um ein BASIC-Programm, wird so verhindert.
Das Einladen und Abspeichern geschieht mit den ganz normalen Lade- und Speicherbefehlen, die ein vorliegendes BASIC-Programm üblicherweise überschreiben. DISK-TO-TYPE aber kann sich retten. Es besteht nämlich aus fünf Teilen: dem BASIC-Programm, Maschinencode in Adressen der Systempages, dem Inhaltsverzeichnis ganz oben im Speicher, den auszuführenden Befehlen im Funktionstastenpuffer und einem weiteren Teil, der die Rettung ermöglicht.

KLIMMZÜGE

Ein SYS-Aufruf auf eine Maschinenroutine legt eine Kopie des BASIC-Programmes ganz oben im Speicher an. Beim Laden wird DISK-TO-TAPE im Regelfall überschrieben, sofern es sich nicht um ein Maschinenprogramm an höherer Adresse oder ein sequentielles File handelt. Im Direkt-

DISK-TO-TAPE

```
100 sys58253:sys58288:poke55,0:pok
e56,246:clr                           <lb>
110 key1,"":key2,"":key3,"":key4,"
":key5,"":key6,"":key7,"":key8,""     <md>
120 fori=312to375                     <kl>
130 reada:pokei,a:next                <gp>
140 data 169,016,162,001,134,043      <ff>
150 data 133,044,133,217,134,216      <nf>
160 data 134,218,162,250,134,219      <nn>
170 data 160,000,162,002,096,032      <ah>
180 data 056,001,169,218,032,148      <ha>
190 data 004,145,216,200,208,246      <jd>
200 data 230,217,230,219,202,208      <kj>
210 data 239,076,075,136,032,056      <ke>
220 data 001,177,216,145,218,200      <fo>
230 data 208,249,230,217,230,219      <ig>
240 data 202,208,242,096              <jh>
250 fori=1015to1074                   <cg>
260 reada:pokei,a:next                <ge>
270 data 104,201,167,208,026,197      <dd>
280 data 216,240,022,169,167,133      <ag>
290 data 216,165,157,133,180,165      <jk>
300 data 158,133,181,169,026,133      <jp>
310 data 043,169,004,133,044,169      <ng>
320 data 167,072,076,101,242,000      <gm>
330 data 000,000,165,180,133,178      <kf>
340 data 165,181,133,179,076,164      <cm>
350 data 241,169,247,141,038,003      <ga>
360 data 169,003,141,039,003,096      <aa>
370 fori=1630to1658                   <ip>
380 reada:pokei,a:next                <he>
390 data 162,001,032,201,255,162      <pa>
400 data 008,032,198,255,032,207      <ld>
410 data 255,170,032,183,255,208      <pi>
420 data 007,138,032,210,255,024      <eo>
430 data 144,240,076,204,255          <hk>
440 sys1064:poke816,29:poke817,4      <od>
450 scnclr:input"kassette nr.";a      <gn>
460 open8,8,8,"cass"+chr$(a+48)+",
s,r"                                  <fl>
470 ax=62992                          <pk>
480 input#8,a$                        <in>
490 fori=1tolen(a$):x$=mid$(a$,i,1
):gosub520:next:x$=chr$(13):gosub5
20                                    <ed>
500 ifa$<>"#"then480                  <ln>
510 close8:goto530                    <di>
520 pokeax,asc(x$):ax=ax+1:return     <lb>
530 poke62990,16:poke62991,246        <fk>
540 key4,"del-550"+x$+"sys358"+x$+
"run"+x$                              <nd>
550 poke2035,3:sys56364:end           <pp>
560 sys58253:sys58288:clr             <kd>
570 a$="":ax=256*peek(62991)+peek(
62990)                                <bd>
580 a=peek(ax):ax=ax+1                <cc>
```

```
590 ifa<>13thena$=a$+chr$(a):goto5
80                                  <ad>
600 ah=int(ax/256):al=ax-256*ah     <ac>
610 poke62990,al:poke62991,ah:x$=c
hr$(13)                             <ap>
620 ifa$="#"then670                 <nk>
630 ifleft$(a$,1)="@"then690        <jj>
640 a$=chr$(34)+a$+chr$(34)         <fp>
650 b$="10"+a$+",8,1"+x$+"sA"+a$+"
,1"+x$+"sys335"+x$+"p04096,0"+x$+"
run"+x$                             <ai>
660 key4,b$:poke2035,3:sys56364:en
d                                   <hj>
670 poke62990,16:poke62991,246:vol
8                                   <oj>
680 fori=400to1000step4:sound1,i,1
:next:sound3,700,50:end             <hd>
690 printa$:a$=right$(a$,len(a$)-1
)                                   <pd>
700 open8,8,8,(a$+",s,r")           <pb>
710 open1,1,1,a$                    <jn>
720 sys1630                         <cm>
730 close8:close1:run               <lj>
```

modus wird neben den zum Kopieren erforderlichen Befehlen aber noch ein SYS-Aufruf bearbeitet. Dieser kopiert das Backup von DISK-TO-TYPE wieder an den BASIC-Anfang, wo es zum Wiederaufruf zur Verfügung steht. Beim dann erfolgenden Start wird nicht etwa dasselbe File neu kopiert, sondern, da ein Zeiger stets für die richtige Abarbeitung sorgt, das jetzt an die Reihe kommende.
Ist alles kopiert, informiert ein Soundsignal darüber. Der Zeiger zum Lesen des Inhaltsverzeichnisses weist wieder auf den Anfang. Beim erneuten Aufruf des Kopierprogrammes werden wieder dieselben Files kopiert. Beim ersten Aufruf existierte noch eine Abfrage, die sich aber nach dem Einlesen des Inhaltsverzeichnisses von Diskette selbst gelöscht hat.
Die zu kopierenden Files sind vor der Inbetriebnahme des Kopierprogrammes in einem sequentiellen File namens CASS0 bis CASS9 zu speichern. Das Kopierprogramm fragt am Anfang nach der Kassettennummer. Die Eingabe einer Zahl von null bis neun veranlaßt den Rechner, CASS0 oder das entsprechende Verzeichnis zu laden. Dieses kann mit Script-Plus oder auch in BASIC erstellt werden mit:

```
OPEN8,8,8,"CASS0,S,W"
PRINT#8,"FILE-BAS"
PRINT#8,"FILE-MC"
PRINT#8,"@FILE-SEQ"
PRINT#8,"#"
CLOSE8
```

BASIC- und Maschinenprogramme werden gleich behandelt. Sequentielle Files sind mit einem Klammeraffen als solche zu markieren. Das Raute-Zeichen signalisiert das Ende des Verzeichnisses. Wer ein Relais bastelt, mit dem sich mehrere Datasetten anschließen lassen, kann in einem einzigen Kopiervorgang gleich einen ganzen Satz von Kassetten herstellen. Bevor nach dem Kopieren wieder von Hand etwas geladen werden soll, ist die Reset-Taste zu drücken, da durch den verstellten Ladezeiger sonst wichtige Systemadressen überschrieben werden. Das würde zu einem Absturz führen. □

Und sie passen doch!

Wie in dem bekannten Lied von den zwei Königskindern, die nicht zusammen kommen konnten, ergeht es einem, wenn man seine Computer an einen Monitor oder ein Fernsehgerät anschließen will. Denn vor das Bild auf der Mattscheibe hat der Hersteller der Geräte einige Fragezeichen gesetzt.

Hat man beim Händler, der einem den Monitor oder Computer verkauft hat, keine Komplettlösung (Computer, Monitor und Kabel) erstanden, wird es recht schwierig, seine Geräte miteinander zu verbinden. Schaut man sich die entsprechenden Rückseiten an, wird einem manchmal schwindelig ob der Vielzahl der verschiedenen Anschlußbuchsen. Da gibt es runde, eckige, trapezförmige und dergleichen mehr. Selbst wenn die Stecker mechanisch zueinander passen, kann die elektrische Verbindung ganz anders sein. Anscheinend handeln manche Hersteller nach dem Motto „Mal sehen, was wir für Stecker auf Lager haben" und bauen diese dann ein. Auch den Anschluß handhaben manche nach Gutdünken. Wir haben uns umgesehen und für Sie zusammengetragen, was es für Stecker und Buchsen an den Geräten gibt.

DER RICHTIGE STECKER

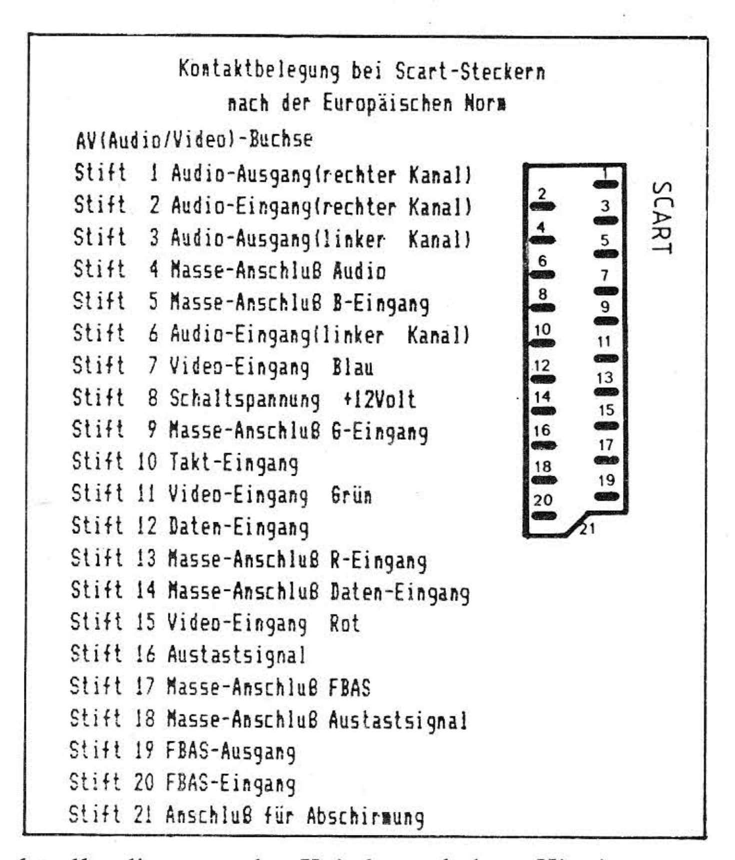
Wenn wir die Commodore-Computer VC-20 oder einen älteren C64 betrachten, fällt auf, daß diese Geräte eine fünfpolige DIN-Buchse besitzen, wogegen die neueren C64er mit einer achtpoligen ausgestattet sind. Diese Buchsen findet man auch am C16/116, Plus/4 und am PC 128 (40 Z.). Wie sie belegt sind, können Sie den Zeichnungen entnehmen.

DIE ANSCHLÜSSE

Auf der Monitorseite sieht die Sache bereits anders aus. Hier gibt es einpolige Cinchbuchsen, sechs- oder achtpolige DIN-Buchsen, neunpolige Sub-D-Buchsen oder achtpolige Honda-Stecker, Scart-Buchsen und was dergleichen mehr auf dem Markt angeboten wird. Hier eine einheitliche Anschlußbelegung zu finden, ist nicht einfach.
Nachstehend finden Sie eine Auswahl der gängigsten Steckverbindungen, die Sie an Ihrem Computer beziehungsweise Monitor finden können. Anschließend haben wir die Verbindungen zusammengetragen, die es ermöglichen, Ihren Commodore-Computer mit Ihrem Monitor zu verbinden.

WIE GEHT'S NUN?

An Ihrem Monitor werden Ihnen die unterschiedlichsten Buchsen auffallen, von denen anscheinend auch die Hersteller nicht genau wissen, wozu sie da sind. Wie sollte man es sich sonst erklären, daß einige Produzenten keine passenden Kabel liefern können, beziehungsweise es nicht für nötig halten, die Anschlußbelegung in den Betriebsanleitungen abzudrucken? Bei der Vielzahl der möglichen Verbindungen werden Sie eventuell Probleme haben, bei Ihrem Computer-


Kontaktbelegung bei Scart-Steckern
nach der Europäischen Norm
AV(Audio/Video)-Buchse
Stift 1 Audio-Ausgang(rechter Kanal)
Stift 2 Audio-Eingang(rechter Kanal)
Stift 3 Audio-Ausgang(linker Kanal)
Stift 4 Masse-Anschluß Audio
Stift 5 Masse-Anschluß B-Eingang
Stift 6 Audio-Eingang(linker Kanal)
Stift 7 Video-Eingang Blau
Stift 8 Schaltspannung +12Volt
Stift 9 Masse-Anschluß G-Eingang
Stift 10 Takt-Eingang
Stift 11 Video-Eingang Grün
Stift 12 Daten-Eingang
Stift 13 Masse-Anschluß R-Eingang
Stift 14 Masse-Anschluß Daten-Eingang
Stift 15 Video-Eingang Rot
Stift 16 Austastsignal
Stift 17 Masse-Anschluß FBAS
Stift 18 Masse-Anschluß Austastsignal
Stift 19 FBAS-Ausgang
Stift 20 FBAS-Eingang
Stift 21 Anschluß für Abschirmung

händler die passenden Kabel zu erhalten. Hier ist Selbsthilfe angebracht. Bei einigen Kabeln dürfte dies nicht so schwierig sein, vor allem, wenn es sich nur um wenige Verbindungen handelt. Einige andere werden es Ihnen nicht so einfach machen. Um zum Beispiel den Amiga an ein Fernsehgerät mit Scartbuchse oder den PC 128 im RGB-Modus an den entsprechenden Monitor anzupassen, bedarf es einiger Kenntnisse der Steckerbelegung beider Geräte.
Für alle anderen Fälle gibt es folgende Möglichkeiten: Sie senden der COMMODORE-WELT-Redaktion die Angaben über Ihre Computer- und Monitortypen (eventuell Fotokopie der Steckerbelegung aus der Bedienungsanleitung) mit einem Freiumschlag zu. Wir werden versuchen, eine Möglichkeit zu finden, wie Sie Ihren Computer mit dem Monitor verbinden können. Wir teilen Ihnen auf Anfrage Adressen von Firmen mit, die sich speziell mit dieser Materie beschäftigen.

DIE RICHTIGE VERBINDUNG

Nachdem das Problem der richtigen Stecker gelöst wäre, kommt das Kabel ins Spiel, nämlich das zwischen den Steckern. Es spielt nicht mal die Zahl der Adern eine Rolle, sondern deren richtige Belegung im Stecker. Nachstehend haben wir die am meisten verwendeten Verbindungen aufgezeichnet. Bei einigen Geräten, auch von gleichen Herstellern, werden große Unterschiede deutlich. Allein beim Anschluß des PC 128 im 80-Zeichen-Modus ergeben sich mehr als fünf verschiedene Anschlußmöglichkeiten, die mehr oder weniger vom verwendeten Monitor abhängig sind.
Bild (1) zeigt die Belegung bei der Verwendung eines RGB-Monitors mit neunpoliger Buchse. Hier spielt der PC 128 natürlich seine volle Leistungsfähigkeit hinsichtlich der Auflösung und der Farbe aus. Dasselbe Ergebnis erreicht auch die Belegung nach Bild (6). Nur wird hier auf der Monitorseite eine achtpolige RGB-Buchse nach DIN 45326 verwendet.

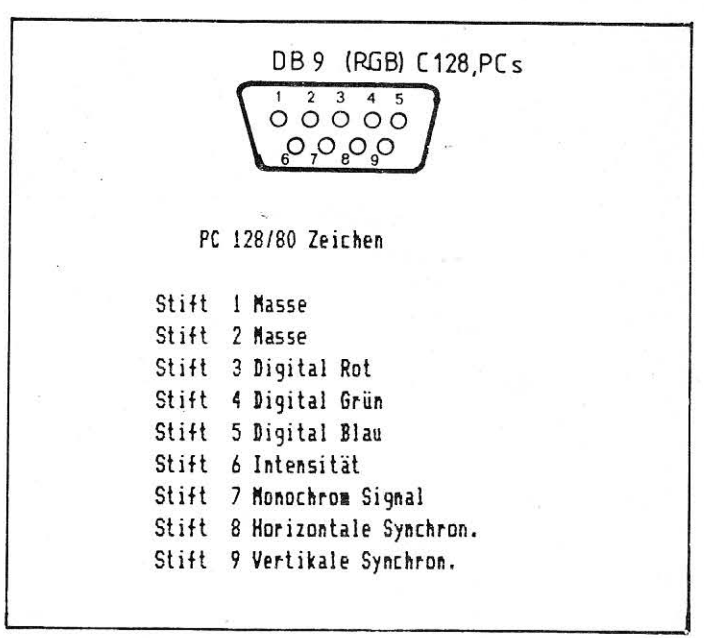

DB 9 (RGB) C128,PCs
PC 128/80 Zeichen
Stift 1 Masse
Stift 2 Masse
Stift 3 Digital Rot
Stift 4 Digital Grün
Stift 5 Digital Blau
Stift 6 Intensität
Stift 7 Monochrom Signal
Stift 8 Horizontale Synchron.
Stift 9 Vertikale Synchron.



Bild 1: Anschluß eines RGB-Digital-Monitor mit neunpoliger Anschlußbuchse an den PC 128 im 80-Zeichen-Modus.

Bild 2: Verbindung des C64/C16/C116/PC 128 und Plus4 über das FBAS-Signal mit einem Monitor (Fernseher) mit sechspoliger DIN-Buchse.

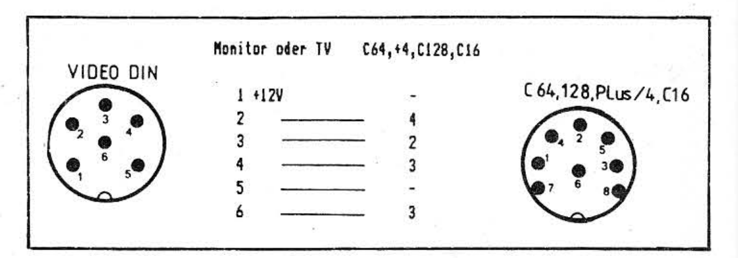


Bild 3: Verbindung des C64/C16/C116/PC 128 und Plus4 über das FBAS-Signal an einem Monitor (Fernseher) mit Scart-Buchse.

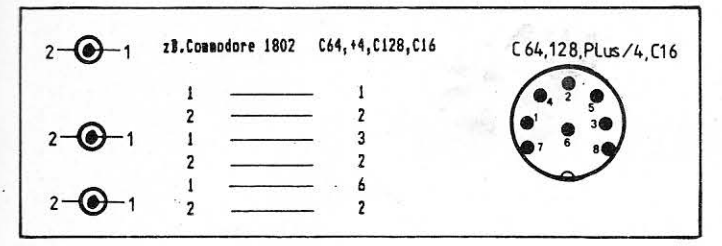


Bild 4: Verbindung des C64/C16/C116/PC128 und Plus 4 an einem Monitor mit getrennten Helligkeits- und Farbsignalen.

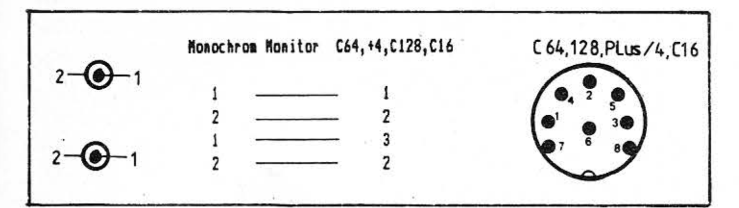


Bild 5: Verbindung des C64/C16/C116/PC128 und Plus4 über das FBAS-Signal mit einem Monitor mit Cinch-Buchsen. Ebenso wird auch ein Monochromer-Monitor angeschlossen.

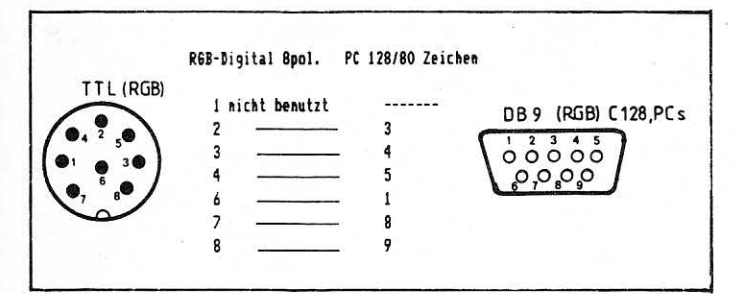


Bild 6: Anschluß eines RGB-Digital-Monitors mit achtpoliger DIN-Anschlußbuchse an den PC128 im 80-Zeichen-Modus.

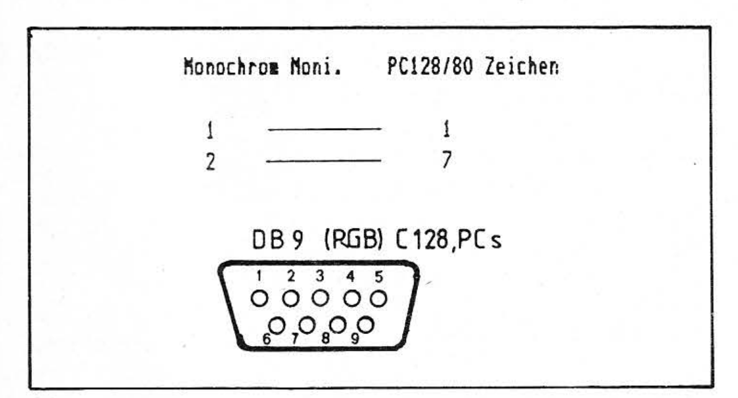
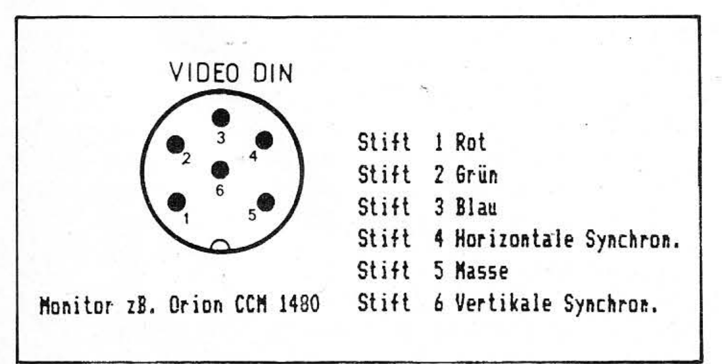
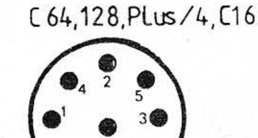


Bild 7: Anschluß eines monochromen Monitors im 80-Zeichen-Modus des PC128.



C 64,128,PLus/4,C16

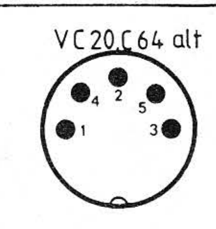


C64,Plus/4,C16/116

Stift 1 Luminanz
Stift 2 Masse
Stift 3 Audio aus
Stift 4 FBAS
Stift 5 Audio ein
Stift 6 Chrominanz
Stift 7 nicht benutzt
Stift 8 nicht benutzt

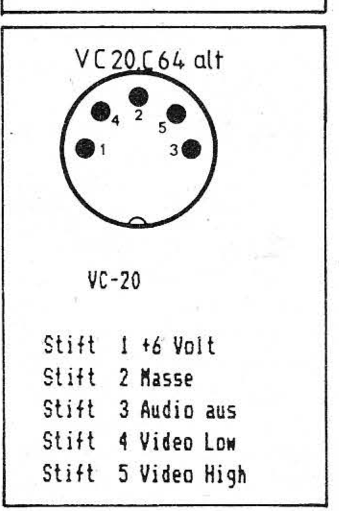


VC 20,C 64 alt



C64 ältere Ausführung

Stift 1 Luminanz
Stift 2 Masse
Stift 3 Audio aus
Stift 4 FBAS
Stift 5 Audio ein



Dabei findet allerdings keine Übertragung des Tonsignals statt. Diese muß über ein zusätzliches Kabel oder durch ein Kombikabel für Bild und Ton erreicht werden. Für letzteres wird wahrscheinlich nur der Selbstbau übrig bleiben.
Besitzt der Monitor (Fernseher) eine Scartbuchse, ist die Übertragung recht einfach. Hier wäre sogar ein Anschluß auf RGB-Basis (80 Zeichen) möglich. Obwohl der PC 128 nur digitale Signale zur Verfügung stellt, können durch Einlöten von Widerständen die Signale gebändigt werden. Den Anschluß für Bild- und Tonsignale im 40-Zeichen-Modus entnehmen Sie dem Bild (3).
Die Bilder 2, 4 und 5 zeigen den Anschluß Ihres Commodore-Computers an Monitore, die mit Cinch-Buchsen ausgerüstet sind. Dies ist häufig bei Original-Commodore-Monitoren der Fall. Alle Belegungen der Bilder 2, 3, 4 und 5 gelten analog auch für die Computer C64/C16/C116 und Plus4. Natürlich muß zwischen einem alten und einem neueren C64 unterschieden werden.
Bild (7) zeigt den Anschluß des 128er an einen monochromen Monitor im 80-Zeichen-Modus.
Eine Besonderheit stellt der Computerklassiker VC-20 dar, weil an der Videobuchse auch noch eine Spannungsversorgung für den Fernsehmodulator anliegt. Hier weicht die Belegung von der sonstigen Commodore-Norm ab. □

Jetzt perfekt: Unser Checksummer

Hatte bisher unser Checksummer an Buchstabenvertauschungen nichts auszusetzen, so zeigt er sich nun nicht mehr so kulant.

Ob Sie mit der alten Version nun eingegeben hatten:

10 print"ab"
oder
10 print "ba",

der Checksummer brachte in beiden Fällen die Prüfsumme < gk > . Leicht kann es vorkommen, daß beim schnellen Tippen, besonders im Zehnfingersystem, die Taste, die eigentlich erst als übernächste drankommen sollte, ein wenig zu früh erwischt wird. Dem Checksummer, der lediglich die Ascii-Werte der Buchstaben addierte, konnte dieses natürlich nicht auffallen. Was also tun? Ob etwas früher oder später addiert wird, ändert nichts am Resultat der Summe. Anders ist es, wenn man zwei Verknüpfungsarten kombiniert. So ist z.B. 2*30+40 etwas anderes als 2*40+30. Und genau dieses war dann die Lösung. Die Summe wird nun einfach durch eine Linksverschiebung vor jeder Addition verdoppelt. Dadurch, daß im Falle, wenn das Ergebnis größer als 255 ist, der dabei entstehende Übertrag als Wert 1 zusätzlich addiert wird, verflüchtigen die Werte der am Anfang der Zeile gefundenen Codes sich nicht nach 8 weiteren Zeichen. Damit bleibt nicht nur die Aussagekraft der Prüfsumme voll erhalten, sondern erfährt sogar eine erhebliche Steigerung. Und vor allen Dingen wird nur eine klitzekleine Änderung erforderlich, die dieses zu vollbringen, in der Lage ist. Ein einziges Byte ist nur zu ändern. Wir tun dieses mit "poke 345,10" in der Zeile 470. Dadurch wird das hier ursprünglich ansässige CLC (Clear Carry) durch ASL (Arithmetik Shift Left) ersetzt. Die nachfolgende Addition mit ADC (Addiere mit Carry) addiert den Ascii-Code des gefundenen Zeichens und den nach links herausgeshifteten Übertrag. Da einige unserer Leser beklagten, daß das Checksummerlisting nachher noch im Programmspeicher stehen würde, haben wir diesem noch mit einem "new" abgeholfen. New bzw. neu ist nun folgendes.

10 print"ab" ergibt die Prüfsumme<jd>
10 print"ba" die Prüfsumme<jf>

Sie brauchen den Checksummer nicht neu einzutippen. Alles, was Sie tun müssen, ist, die Zeile 470 anzufügen. An der Bedienung des Checksummers hat sich nichts geändert. Die Eingabehinweise bleiben daher wie gehabt.

EINGABEHINWEISE

Am rechten Rand jedes Listings, jeweils am Ende einer Eingabezeile, finden Sie zwei Buchstaben zwischen einem Kleiner- und einem Größerzeichen eingeschlossen. Diese dürfen Sie nicht mit in Ihr Listing eintippen, sondern sie dienen Ihnen zur Überprüfung Ihrer Eingabe.

Zwischen dem Kleiner- und dem Größerzeichen am rechten Rand befinden sich zwei Buchstaben. Mit einem speziellen Programm können Sie beim Eintippen Ihre Eingabe auf ihre Richtigkeit überprüfen. Dieses Programm, der Checksummer, sorgt nämlich dafür, daß nach erfolgter Zeileneingabe am linken oberen Bildschirmeck zwei Buchstaben ausgegeben werden. Wenn diese Buchstaben nicht mit den vorher erwähnten Buchstaben in unserem Listing übereinstimmen, so können Sie davon ausgehen, daß Sie sich vertippt haben und können sich so die Zeile nochmals näher ansehen, ob Sie Ihren Eingabefehler finden. Wenn Sie dann alles richtig getippt haben, so stimmen die Buchstaben überein und Sie können sich getrost der nächsten Zeile zuwenden.

ERST SICHERN, DANN AUSPROBIEREN

Das Checksummerlisting hat noch keine Prüfsummen. Seien Sie deshalb besonders aufmerksam, daß alles paßt und speichern Sie dieses Programm unbedingt ab, bevor Sie es starten! Bei einem Tippfehler würde es sich wahrscheinlich auf Nimmerwiedersehen verabschieden und Sie müßten die ganze Arbeit vermutlich nochmals machen. Wenn Sie es gestartet haben, so geschieht nichts Besonderes. Der Computer meldet sich einfach kurz darauf mit „READY", und das war auch schon alles. Alles sollte nun wie immer funktionieren, mit der kleinen Ausnahme, daß nunmehr nach jeder Eingabe im Direktmodus eine Prüfsumme erscheint. Nehmen Sie zum Testen irgendeine kurze Basiczeile aus unserem Heft her und testen sie aus. Wenn die Summen übereinstimmen, so können Sie sich freuen, denn Fehler beim Abtippen werden Ihnen nun in Zukunft viel weniger passieren, als vorher.

EINER FÜR ALLE, EIN ECHTES UNIVERSAL-PROGRAMM

Unseren Checksummer können Sie verwenden, ob Sie einen C16/116/Plus4 oder ob Sie einen C64 oder gar einen C128 haben. Nur müssen Sie beim letzteren beachten, ob Sie auch wirklich im 40-Zeichenmodus sind. Nachdem Sie den Checksummer geladen und gestartet haben, können Sie Ihr Basicprogramm eingeben wie gewohnt, Sie können es abspeichern, Sie können auch laden, Sie können Kürzel verwenden und, ob Sie ein paar Leerzeichen mehr oder weniger verwenden, der Checksummer läßt sich dadurch nicht aus der Fassung bringen. Ein bißchen Vorsicht sollte man allerdings walten lassen, wenn man Programme eingetippt hat, in denen Peeks und Pokes vorkommen. Es wird zwar nicht besonders häufig vorkommen, aber es könnte bisweilen ge-

```
10 rem =checksummer==c16 c64 c128==
20 rem (p) 05/87 commodore welt   ==
30 rem ============================
40 rem (c) alfons mittelmeyer     ==
50 rem                            ==
60 rem c16/116/plus4              ==
70 rem c64                        ==
80 rem c128 (40-zeichen)          ==
90 rem ============================
100 rem ---------------------------
110 rem   grundroutine (c16)
120 rem ---------------------------
130 data165,059,072,165,060,072,032
140 data086,137,104,133,060,104,133
150 data059,152,072,160,000,165,020
160 data024,101,021,170,024,144,011
170 data201,032,240,006,138,024,113
180 data059,234,170,200,177,059,234
190 data208,240,169,031,072,138,074
200 data074,074,074,072,138,041,015
205 data072,169,031,072,162,003,104
210 data024,105,129,157,000,012,202
220 data016,246,104,168,096
230 lt=peek(772):ht=peek(773)
240 fori=312to386:readx:pokei,x:nex
t
250 iflt<>124then350
260 rem ---------------------------
270 rem         anpassung c64
280 rem ---------------------------
290 fori=312to317:pokei,234:next
300 fori=321to326:pokei,234:next
310 fori=1to6:readad:readx:pokead,x
:next
320 poke380,4:poke319,lt:poke320,ht
:goto430
330 data346,121,347,000,348,002
340 data351,185,352,000,353,002
350 iflt<>13then430
360 rem ---------------------------
370 rem anpassung c128 (40 zeichen)
380 rem ---------------------------
390 restore410:poke332,22
400 poke335,23:goto310
410 data313,061,316,062,323,062
420 data326,061,347,061,352,061
430 poke772,056:poke 773,1
440 rem ---------------------------
450 rem ergaenzung 10/87
460 rem ---------------------------
470 poke 345,10:new
480 rem ===========================
490 rem = fuer hefte cw 7/87 bis   =
500 rem = cw 9/87 sowie cw128 5/87=
510 rem = und c16 6/87 ist die     =
520 rem = poke-anweisung in zeile =
530 rem = 470 wegzulassen          =
540 rem ===========================
```

schehen, daß nach dem Laufenlassen eines Programmes weder der Checksummer noch sonst etwas mehr funktioniert, auch wenn dies bisher ohne Checksummer nicht der Fall gewesen sein sollte. Also bitte sichern Sie in jedem Falle Ihre Programme, bevor Sie sie ausprobieren.

Ein paar Dinge sollten Sie noch wissen. Wir drucken in unseren Listings des öfteren Punkte statt Leerzeichen. Wenn Ihnen nun aber Leerzeichen besser gefallen, so liefert der Checksummer natürlich eine falsche Summe. Wenn Sie diese Richtigkeit überprüfen wollen, so können Sie dies tun, indem Sie sie zuerst einmal so wie im Heft abtippen, und nachher, nachdem Sie sie nachgeprüft haben, einfach wieder die Punkte durch Leerzeichen ersetzen.

A. Mittelmeyer

MONITOR

CHECKMON

```
40 rem checkmon ===============c16 <cn>
50 rem (p) commodore welt team     <ke>
60 rem ========================== <nk>
70 rem (c) by  a.mittelmeyer       <ag>
80 rem c16/116/plus4               <ki>
90 rem ========================== <jg>
100 fori=312to398:reada            <ei>
110 pokei,a:next                   <ep>
120 data 132,218,108,219,0,132,219 <oe>
130 data 164,218,76,75,236,201,62  <nk>
140 data 208,249,165,161,10,101    <jc>
150 data 162,160,7,10,113,161,136  <ej>
160 data 16,250,133,216,169,30,133 <oh>
170 data 217,169,62,160,97,208,220 <mk>
180 data 198,217,208,218,160,105   <ai>
190 data 208,212,201,13,240,4,164  <ha>
200 data 218,24,96,169,60,160,68   <lh>
210 data 32,61,1,165,216,32,16,251 <ec>
220 data 169,62,160,5,208,2,169,32 <om>
230 data 32,75,236,136,208,248,169 <ol>
240 data 13,208,176,219,68,220,1   <go>
250 data 804,56,805,1              <hn>
260 fori=1to4:reada:readb:pokea,b  <lm>
270 next:new                       <ji>
280 rem ========e=n=d=e=========== <cc>
```

"CHECKMON" ist eine unerlaessliche Hilfe zur Eingabe von Maschinenprogrammen. Laden und starten Sie "Checkmon" und gehen dann mittels MONITOR in denselben. Wenn Sie sich nun z.B. mit 'M1000' einen Speicherbereich ansehen, oder Hexzahlen eingeben, so erscheint rechts die Pruefsumme anstatt der Asciicodes.

C16/P4-USER! Orig.-Software-Rest. Liste gg. Freiumschlag. Nur Orig.-Kass., aber spottbillig. Harald Scheel, Kollaustr. 178, 2000 Hamburg 61

C16/116/P4! HEX-Tastatur f. die komfortable Eingabe v. MC-Programmen m. TEDMON od. Checksummer. In BASIC als 2x16 Funktionstasten verwendbar. DM 60,–. Hans-D. Veit, Rennweg 21, 8441 Aiterhofen

SUPER-ANGEBOT: STAR GAMES 60 zum C16/116/ Plus4 für 39,– Sfr od. 39,– DM. Auf 4 Kass. finden Sie 60 interessante Spiele. Lieferung erfolgt per Nachn. Kostenlose Info bei: M. Greifenhagen, Stöckelstr. 8, CH-8610 Uster

*** SYSTEMWECHSEL *** Plus 4 f. 100,– DM; Floppy 1551 f. 100,–; Drucker MPS 801 f. 250,– DM; Monitor 150,– DM; 2 Joyst. 20,– DM; 3mal M+T-Disk u. Sonderhefte 75,– DM; Lightpen 40,– DM; Script+ 40,– DM; ELCAD u. HiRes-HC 65,– DM; 16C-Welt 25,– DM. K. Gaisser, 07191/44211

TESTWELT sucht noch Mitglieder. Wahlweise Disk od. Tape. Erscheinen monatlich. Bei Fragen bitte wenden an: Jan Peter Kruse, Max-Beckmann-Str. 23, 2900 Oldenburg 23, Ortsteil Kreyenbrück. Nur C16/116 u. Plus4!

COMMODORE USER CLAN IN DER DDR sucht für seine C16/116/Plus4-Freaks Computerzeitschriften, Bücher, Kopien sowie Hardware aller Art u. Kassettenschrott, da keine Verwandten in der BRD. Wir bedanken uns recht herzlich. Ludwig Steuner, Teutoniaweg 1, DDR 9272 Gersdorf

VERKAUFE Orig.-Kass. f. C16: Battle, Cave Fighter, Finders Keepers, L. Tennis, Fing. Malone, Ghost Town, Hektik, Kikstart, Oblido, Olympiad, Petch, Squirm, Vegas Jackpot, Powerball, Spectipede a 5,– DM. Kaiser, Gorch-Fock-Str. 7, 2057 Reinbek, Tel. 040/7104765

C16/P4-Kopiermodul, Anwender u. Spielprog.! Super-Gratisinfo m. Tips u. Tricks (Poke-Liste) sofort anfordern bei: Th. Görtz, Friedrich-Ebert-Str. 113, 6103 Griesheim

SUCHE preisgünstig wengi gebr. Floppy 1541 sowie Traktordrucker f. P4 (kein Thermo), ältere Hefte P4/ C64, verbesserte Datenverw.-Programme f. P4. Marian Demowski, Dr.-Fitz-Str. 10, 8832 Weißenburg

C64 – C16 – Plus4 – Lernprg. Techn. Mathe + Schulanwend. + Grafik zu reellem Preis. Cass/Disk z.B. Bruchrechn. Vokab. Geometrie, Zahnr. Festigk. Hydr. E-Techn. Katalog 1 DM Briefm.-Comp. Typ angeben! Softvers. A. Ristau Peetzweg 9, 3320 Salzgitter 1

PLUS4-USER AUS DER DDR sucht 1551 od. 1541 billig od. geschenkt. Habe auch viel Software (Disk u. Tape). Dirk Schäfer, Junkerstr. 32, DDR-5900 Eisenach

LOGO für den Plus4, C16 (64K), Modul, Diskette u. Handbuch f. DM 50,– bei Vorkasse (NN: plus DM 5,–). Div. Software auf Steckmodul. Info gg. Freiumschlag. Harald Hobbelmann, Junkernkamp 18, 2822 Schwanewede, Tel. 04209/5390

C16/P4-Datenverwaltung: 170 KB, bis zu 5000 Daten fest im Griff, hohe Datensicherheit, bedienungsfreundlich, superschnell! Gratisinfo bei: Th. Görtz, Friedrich-Ebert-Str. 113, 6013 Griesheim

Wer würde mittellosem DDR-Studenten einen Drucker f. Plus4 schenken? U. Krause, Leninstr. 20/43, DDR-8300 Pirna 2

PLUS 4 C – 16
80 Zeichen Textverarb.
deutsche Umlaute
Disk DM 39,90 Fa. BLK
Kaspar-Spät-Str. 15
8000 München 90,
Telefon 089 / 68 82 26

GRAFIKPROGRAMME! Wer tauscht mit mir Grafikprogramme? Tel. 07031/ 806342. Oder schreibt an: Daniel Bacher, Dachsklinge 5, 7032 Sindelfingen

UMFRAGE! Ich mache eine Umfrage bei den P4/C16/116-Besitzern, welches das beste Spiel für die o.g. Computer ist. Einfach anrufen unter 04838/965 u. es nennen.

SUCHE alte Computer-Zeitschriften (CW, Happy Computer, 64er usw.). Benötige unbedingt Happy Computer Nr. 1/86! SEHR WICHTIG! Angebote an: Throsten Boese, Pappelweg 23, 2300 Kiel 1

SUCHE VC 1520 Plotter/ Printer. Wo bekomme ich die 4-1/2-Zoll-Additionsrollen? Paul Brune, Schreinerstr. 4, Tel. 05242/34689

TAUSCHE Software f. Plus4. Biete ca. 350 Games u. 100 Anw. Jede Zuschrift wird beantwortet. Schickt Eure Listen an Thomas Weisheit, Pommrück 6, 6431 Hauneck

VERKAUFE Liste m. 35 Pokes u. 50 Startadressen v. bekannten Spielen f. C16/116/ P4 f. nur 10,– DM-Schein an: Jens Borau, Ahornstr. 4, 3549 Volkmarsen

C16/P4: VERKAUFE 6 Orig. Spiele (Tape) f. 50,– DM. 2 Spiele nur f. 64 KB. Tel. 05674/4645

Wer hat DFÜ-Erfahrungen m. Plus4? Bitte melden bei Rainer Bielefeld, Neustadtring 23, 3300 Braunschweig, Tel. 0531/503589

PLUS4/C16. Orig.-Spiele (Disk) zu verk.: Profipack II 64K 80,– DM; Music Master, Grafikdesigner, Micro Text 2.0 je 10,– DM; Plus4 Sextett 20,– DM; alles zzgl. Porto. Heinz Weissmann, Im Leitle 2, 8570 Pegnitz, Tel. 09241/ 2627

VERKAUFE 64er-SH 14 (C16 etc.) m. Prog.-Disk f. 20,– DM; CW-SH 4/87 f. 6,50 DM; Comp. mit SH 1/87 f. 6,50 DM; Comp. mit SH 1/87-3/87 a 3,80 DM; Comp. mit 12/86-3/87 a 1,70 DM; 64er 12/86, 2/87, 3/87, 6/87 a 3,80 DM. Kaiser, Gorch-Fock-Str. 7, 2057 Reinbek, Tel. 040/7104765

Gehäuse f. Plus4, C16 f. DM 20,–. Div. Software auf Eprom. Viele Ersatzteile f. C16 u. Plus4. Info gg. Freiumschlag. Harald Hobbelmann, Junkernkamp 18, 2822 Schwanewede, Tel. 04209/5390

SUCHE Floppy 1541 od. 1551 u. Farbmonitor od. Farbfernseher. Nur funktionstüchtig. Zahle je bis zu 150,– DM. Tel. 02261/ 48940 ab 17 Uhr

HAST DU PROBLEME m. dem C16/Plus4? Du bist Anfänger? Wir helfen. Auch Fortgeschrittenen. Wir bieten Clubzeitung, Infos, jede Menge Tips u. Tricks. Am besten heute noch Info gg. 1,60 DM bei CIG, c/o K.-D. Schindler, Luciusstr. 10, 623 Frankfurt 80, anfordern.

PLUS4, Floppy 1551, Datasette, Monitor, Joyst., div. Bücher u. Zeitschr., Anwenderpr. z.B. Micro Text, Datei, Kalk usw., Spiele z.B. ACE, Bongo, Demol., Strip Poker usw. wg. Systemwechsel zu verk. VB 400,– DM. Lars Winter, Schloßgebiet 4, 2320 Plön, Tel. 04522/ 3753, 17-18 Uhr

Wer kann mir helfen? Suche komplette Liste, um selber Programme u. Spiele zu schreiben! Bin quasi noch Anfänger! Unkosten werden übernommen! Interesse für C16 m. 64K u. C64. Wer kann mir helfen? Hans Jürgens, Provinzstr. 111, 1000 Berlin 51

AUFGEPASST! Verkaufe Plus4, Datasette, 1 Joy, 11 Comp.-Hefte, 22 Spiele, 3 Handbücher, 6 Mon. alt, Preis nach VB od. Tausch gg. C64. Tel. 0201/466338

C16/116/Plus4. Entwicklung v. Rechenprogrammen jeder Art. Statistik u. Kalkulation! Martin G. Maaß, Kulbrockstr. 2, 4800 Bielefeld 14

VERKAUFE f. C16/Plus4 Spiele u. Anwenderprog., 20 Prog. zu 10,– DM. VERKAUFE C16-Orig.-Spiele zum halben Preis z.B. P.O.D, R16, Attack. Schreibt an: C. Schnoor, Lausanner Str. 129, 2800 Bremen 44

VERKAUFE: C116/C16/ Plus4. Anschlußkarte f. Tastatur m. 10er Block (CBM 600/700), Erw. Karte m. 2 Steckplätzen f. Expans.-Port, z.B. f. Eproms. Brenne Ihre Eproms. Suche Hard- u. Software aller Art. Tel. 040/ 7125694

Plus4/C16/C116: Biete 5C Spiele, div. Anwend.-Prog. f DM 20,– (Disk). Schein od Scheck. Suche Kopierprog. v Kass. auf Disk. Listings gg 80 Pf. in Briefmarken.

C16/116 P4 — Speicherbegrenzung auf 12 K

Speicherbegrenzungen per Software

Manche Software läuft nur auf Rechnern mit zwölf KByte Speicher. Der Speicherbereich muß auf zwölf KByte begrenzt werden.

1. Methode: (Bei BASIC-Prg):
POKE55,246:POKE56,63:CLR

2. Methode:
POKE1331,246:POKE1332,63:SYS32768
Bei 32 KByte:
POKE1331,246:POKE1332,127:SYS32768

C16/116 P4 — Tastencode (2038)

Letzte gedrückte Taste

Nach jedem Tastendruck steht in der Speicherstelle 2038 der Tastencode der zuletzt gedrückten Taste. Ist keine Taste gedrückt, ist der Inhalt 0. Mit dem WAIT-Befehl kann man dies nutzen:

WAIT2038,63: Programm wartet auf beliebige Taste.

WAIT2038,60,60: Programm läuft so lange, bis der Inhalt von 2038 genau 60 ist (Leertaste).

C16/116 P4 — Merkzettel Bildschirm

Bildschirminhalt abspeichern

Man kann sich einen einfachen Zettelkasten auf Diskette (Kassette) anlegen:

1) Bildschirm löschen

2) Text schreiben (maximal halber Bildschirm)

3) Im MONITOR mit „S“name“,8,0C00,0D00“ abspeichern

4) Kann später im MONITOR mit „L“name“,8“ geladen werden.

C16/116 P4 — GOTO ZN

Programmierter Zeilensprung

Die Sprungbefehle GOTO und GOSUB akzeptieren keine Variablen im Argument. Hierzu ist der ON-Befehl gedacht:

Beispiel (Menü):
100 v$="1234":getkeya$
110 on instr(v$,a$)+1 goto100,200,300,400,500

Ist A$ nicht in V$ enthalten, springt das Programm nach 100, ist A$="4", springt es nach 500.

C16/116 P4 — Interruptvektor (788/789)

Verbiegen des IRQ

Normalerweise zeigt der IRQ-Vektor (steht in 788/789) auf $CE0E. Will man ein interruptgesteuertes Programm starten, so muß dieser Vektor auf das eigene Programm umgelenkt werden.
Die eigene Routine muß mit JMP $CE0E abgeschlossen werden und es darf während der Umstellung des Vektors kein Interrupt erfolgen. Dies geschieht mit einer kleinen Routine (siehe Rückseite).

C16/116 P4 — Einfügemodus (2026)

Anzeige des Einfügemodus

Mit ESC&A wird der Insert-Modus ein- und mit ESC&O wieder ausgeschaltet.

Die dafür zuständige Speicherstelle ist 2026 ($07ea, $00 = aus, $80 = ein).
Mit einem kleinen Maschinenprogramm (auf der Rückseite) kann man sich dies zunutze machen. Es zeigt durch die Rahmenfarbe (steht in $ff19) an, ob der Einfügemodus eingeschaltet ist.

C16/116 P4 — ESC aktiv? (2027)

ASCII-Code des letzten Zeichens

In 2027 ($07eb) steht der ASCII-Code des letzten Zeichens. Bei ESC steht dort 27 ($1b). Damit kann man sich anzeigen lassen, ob ESC aktiv ist (Maschinenprogramm auf der Rückseite).
Dies ist vor allem dann hilfreich, wenn die Tastatur schon stark beansprucht ist und die ESC-Taste nicht mehr immer anspricht. Man kann auf diese Art natürlich auch andere Zeichen abfragen (Menüauswahl bei Maschinenprogrammen).

C16/116 P4 — Externe Variablendeklartion

Hochsetzen des BASIC-Anfangs

Durch Hochsetzen des BASIC-Anfangs (in 43/44) kann man vor dem BASIC-Programm ein Maschinenprogramm ablegen.

Wird der Beginn des BASIC-Speichers (in 43/44) verändert, so hat dies keinen Einlfuß auf die Variablen, die hinter dem BASIC-Programm abgelegt werden. Daher kann man ein BASIC-Programm in einen Teil, der Variablen definiert, und den Hauptteil auftrennen (wird schneller, da kürzer).

Demo Tastencode

```
100 v=63:wait2038,v:geta$:printa$peek(2038):goto
      100:rem laeuft bei bel.taste
200 v=60:wait2038,v,v:printpeek(2038),peek(2038)
      andv:goto200:rem haelt m. leert.
```

Beste Methode:

```
49998   rem ** speicher auf 12 k   ****
49999   rem *****    resetfest ******
50000   restore50020:fori=16374toi+7:readx:pokei,x:next
50010   poke1331,246:poke1332,63:sys32814:return
50020   data 141,62,255,76,164,242,246,255
```

Mit GOSUB50000 starten, mit POKE16377,1 und Drücken des RESET-Knopfes die Begrenzung aufheben.

GOTO ZN mit TRAP

Bei einem Fehler springt das Programm an die mit TRAP angegebene Zeilennummer. Das Argument kann hier eine Variable sein.

Beispiel:

```
10 a=900+int(rnd(1)*4+.5)*10:trap a:*
```

„*“ erzeugt einen SYNTAX ERROR und das Programm springt (zufallsgesteuert) zu den Zeilen 910, 920, 930 oder 940.

Beispiel:

Nach CLEAR eingeben:

POKE4097,1:SYS34840:SYS34891:CLR

Im MONITOR:

S“RENEW“,8,0C00,0C50

Wenn jetzt ein Programm durch RESET verloren geht, einfach LOAD“RENEW“,8,1 eingeben, mit HOME und RETURN drücken und das Programm ist wieder da. (Bei Kassette statt der Acht eine Eins schreiben.)

Rahmenfarbe zeigt Insert-Modus an:

```
. 065e   lda   $07ea
. 0661   bne   $066b
. 0663   lda   $01
. 0665   sta   $ff19
. 0668   jmp   $ce0e
. 066b   lda   $00
. 066d   sta   $ff19
. 0670   jmp   $ce0e
```

Der IRQ-Vektor muß auf $065E umgelenkt werden (POKE788,94:POKE789,6 oder besser als Maschinenroutine).

Beispiel:

Interrupt-Vektor auf $065e verbiegen.

```
. 0675   sei
. 0676   lda   #$5e
. 0678   sta   $0314
. 067b   lda   #$06
. 067d   sta   $0315
. 0680   cli
. 0681   rts
```

Natürlich muß vor dem SYS-Aufruf zum Vektorumstellen (SYS1651) die eigene Routine bei $065e vorliegen.

Beispiel:

Teil 1 (Deklaration):

```
10 poke44,20:a$=“BASIC-Anfang bei 20*256“
20 goto 10
```

Jetzt BASIC-Anfang im Direktmodus hochsetzen mit:
POKE44,20:POKE20*256,0

Teil 2 (Hauptprogramm):

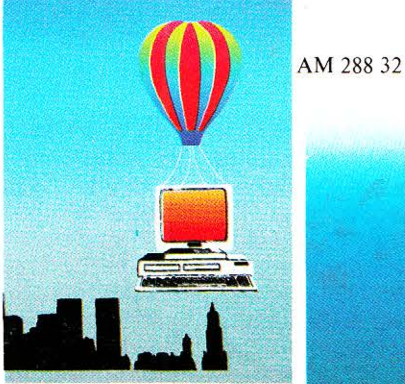
```
10 print a$:rem
```

Mit POKE44,16 BASIC-Anfang zurücksetzen und mit RUN starten. In Teil 1 (Zeile 10) wird der BASIC-Anfang auf den Beginn des Hauptprogramms gesetzt; der C16 merkt dies aber erst, wenn eine GOTO-Anweisung kommt. Dann sucht er die Zeile 10 ab dem (neuen) BASIC-Anfang.

Rahmenfarbe (in $FF19) ändert sich, wenn ESC gedrückt wurde:

```
. 065e   lda   $07eb
. 0661   cmp   #$1b
. 0663   bne   $066d
. 0665   lda   $00
. 0667   sta   $ff19
. 066a   jmp   $ce0e
. 066d   lda   $01
. 066f   sta   $ff19
. 0672   jmp   $ce0e
```

Der IRQ-Vektor muß auf $065E umgelenkt werden (POKE788,94:POKE789,6 oder besser als Maschinenroutine).

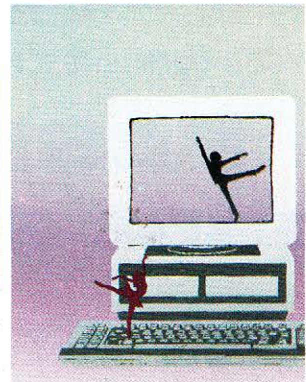


AM 288 31



AM 288 32

AM 288 33



AM 288 34

AM 288 35

AM 288 29



AM 288 30

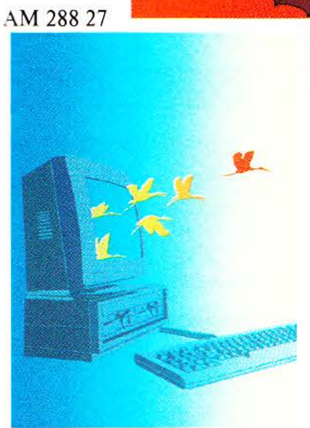
Edition Computer-Kunst



Computer sind nicht nur Gebrauchsgegenstände. Sie werden zunehmend auch von Künstlern als Motiv entdeckt. Einige besonders gut gelungene Arbeiten haben wir für Sie, unsere Leser, reservieren können. Alle Motive sind strikt auf eine Auflage von 99 Exemplaren limitiert und von der Künstlerin, Sybille Areco, handsigniert und numeriert. Die Exponate werden nach Bestelleingang im 24-Farbendruck von Hand gefertigt, die Vorlage nach dem 99. Druck vernichtet. Unser Angebot: Jedes Motiv nur DM 85,–, zwei Motive DM 150,–, drei DM 210,– und vier Motive nur DM 250,–. Jedes Bild ist 30 x 40 Zentimeter groß und kommt im Passpartout in stabiler Verpackung (im Preis enthalten).

Lieferzeit nach Bestelleingang: ca. drei Wochen.

AM 288 27



AM 288 28



AM 288 25



AM 288 26

Bestellcoupon

Hiermit bestelle ich in Kenntnis ihrer Verkaufsbedingungen folgende Exponate:

Nr.: ______________________________

____

Ich zahle: (Zutreffendes bitte ankreuzen!)

per beigefügtem Scheck ☐ Schein ☐ Gegen Bankabbuchung am Versandtag ☐

Meine Bank (mit Ortsname) ______________

Meine Kontonummer ______________

Meine Bankleitzahl ______________ **(steht auf jedem Bankauszug)**

Nachname ______________ **Vorname** ______________

PLZ/Ort ______________ **Str./Nr.** ______________

Verkaufsbedingungen: Lieferung nur gegen Vorkasse oder Bankabbuchung.
Wichtig: Scheckeinreichung und Bankabbuchung erfolgen erst nach dem Versand. Keine Nachnahme möglich. Auf Wunsch Rechnung mit ausgewiesener Mehrwertsteuer.

Unterschrift ______________

Bitte ausschneiden und einsenden an

AKTUELL-VERLAG
Heßstraße 90
8000 München 40

SO PROGRAMMIERE ICH IN BASIC

BASIC ist ganz einfach

Obwohl fast jeder C16/Plus4-Besitzer erste Erfahrungen mit BASIC vorzuweisen hat, herrscht oft hinsichtlich der BASIC-Programmierung viel Ratlosigkeit. Unsere neue Artikelreihe zeigt auf, wie BASIC für die persönlichen Belange zu nutzen ist.

Falls Sie sich auch schon mal die Frage gestellt haben, was Sie mit Ihrem C16, C116 oder Plus4 anfangen sollen und zu keiner schlüssigen Antwort gekommen sind, stehen Sie damit sicher nicht allein. Die Motivation, sich für Geräte dieser Kategorie zu entscheiden, liegt oft allein am günstigen Preis oder hat den Grund, mal eben zu lernen, wie man mit einem Computer umgeht.

Die ersten Wochen nach dem Erwerb ist so ein Rechner auch sehr interessant und unterhaltsam. Nachdem aber das 10000. Raumschiff abgeschossen und der obligatorische BASIC-Kurs von Kurt Scharnbacher durchgeackert wurde, stellt sich oft genug die Frage: „Was soll ich damit?"

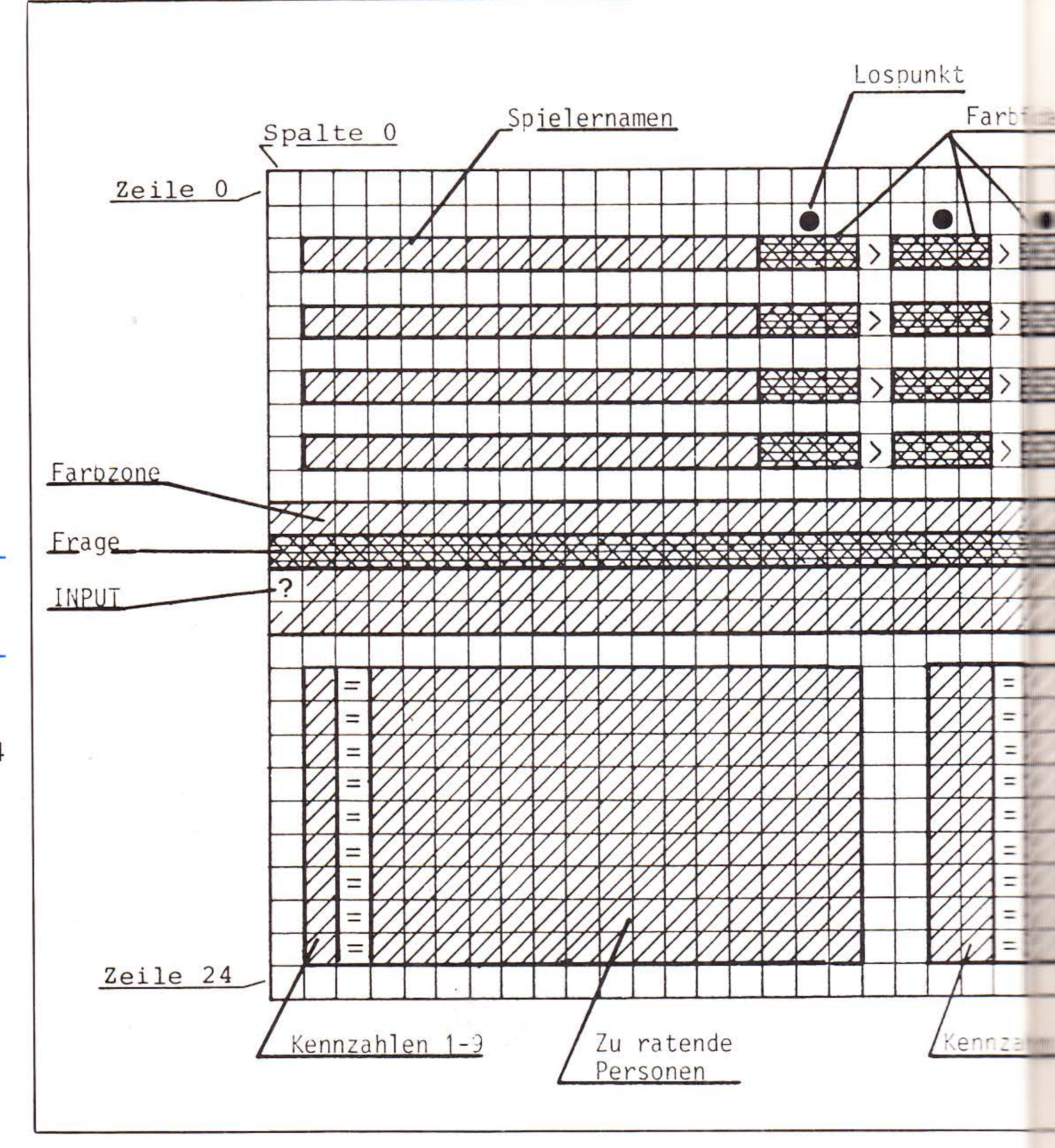
SPIELECOMPUTER ODER ECHTES ARBEITSGERÄT?

Erklärt wird der Frust oft damit, daß man meint, der C16 habe viel zu wenig Speicherplatz, der C64 oder der Amiga würden den nötigen Spaß bringen und daß der C16 nur als Spielzeug zu gebrauchen wäre. Oder daß man nur in Maschinensprache „richtig" programmieren und, last not least, man von einem Wühltischcomputer halt nicht viel erwarten könne.

Das erste Argument ist zwar bedingt richtig, kann aber vergessen werden, wenn man bedenkt, daß sich der C16 mit etwa 50 Mark auf 64 KByte RAM aufrüsten läßt und damit über mehr Speicherplatz verfügt als der C64.

Die Tatsache, daß der C16 und seine „Ableger" C116 und Plus4 als Wühltischcomputer laufen, sagt nichts über die Qualität und Fähigkeiten dieser Rechner aus. Den ersten Computer schaffte ich mir 1982 gebraucht zu einem Preis von 700 Mark an. Es handelte sich um einen TI99, der damals stolze 2000 Mark kostete und nur magere 16 KByte RAM vorzuweisen hatte. Er hätte dem Plus4 in keiner Hinsicht das Wasser reichen können, trotz des zehnfachen Preises. Vor vier Jahren, als der Plus4 eingeführt wurde, kostete er mit 1200 Mark immerhin 400 Mark mehr als der C64 zur selben Zeit. Er war sogar als möglicher Nachfolger in der Diskussion, denn schließlich ist er wie der C16 mit einem wesentlich verbesserten BASIC und dem komfortableren und schnelleren Disketten-Betriebssystem ausgestattet. Zu welchen Leistungen der C16/Plus4 fähig ist, dokumentieren Programme wie Inter-Olympiade oder Ace.

Auch die „Spielzeug-These" hat wenig Substanz. Unbestritten sind die Computer dieser Reihe gute Spielzeuge. Bei einem Fundus von zirka 300 kaufbaren Programmen und etlichen Publikationen dürfte für jeden Geschmack etwas da-



bei sein. Es hindert Sie jedoch nicht daran, Ihren Rechner auch für „ernsthafte" Anwendungen zu nutzen. Schließlich hat der Plus4 eingebaute Programme, die Ihren Computer im Handumdrehen zur elektrischen Superschreibmaschine machen und die auch mit der Katalogisierung Ihrer Mineralien- oder Briefmarkensammlung spielend fertig werden.
Auch für den C16 kommen in Verbindung mit entsprechender Software derartige Anwendungen in Betracht. In diesen Fällen sollten allerdings ein Drucker und ein Diskettenlaufwerk vorhanden sein. Die Möglichkeiten, die sich aus dieser Kombination ergeben, erschöpfen sich erst im Zusammenhang mit Geschäfts- oder Betriebsanwendungen.
Aber auch ohne Zusatzgeräte, Datasette ausgenommen, finden sich noch genug Möglichkeiten. Vor allem, wenn die geplante Anwendung außerhalb der Standardprogramme (Textverarbeitung, Kalkulationen und Dateien) liegt, steht nichts im Wege, eigene Problemlösungen zu programmieren.

IST BASIC UNEFFEKTIV?

Spätestens dann stellt man sich die Frage: „Mit BASIC könnte man notfalls noch fertig werden, nur – ist das nicht sehr uneffektiv?"
Wer sich nicht vorstellen kann, daß der C16/Plus4 für eigene Anwendungen in Betracht kommen könnte, sollte sich die folgenden Fragen durch den Kopf gehen lassen:

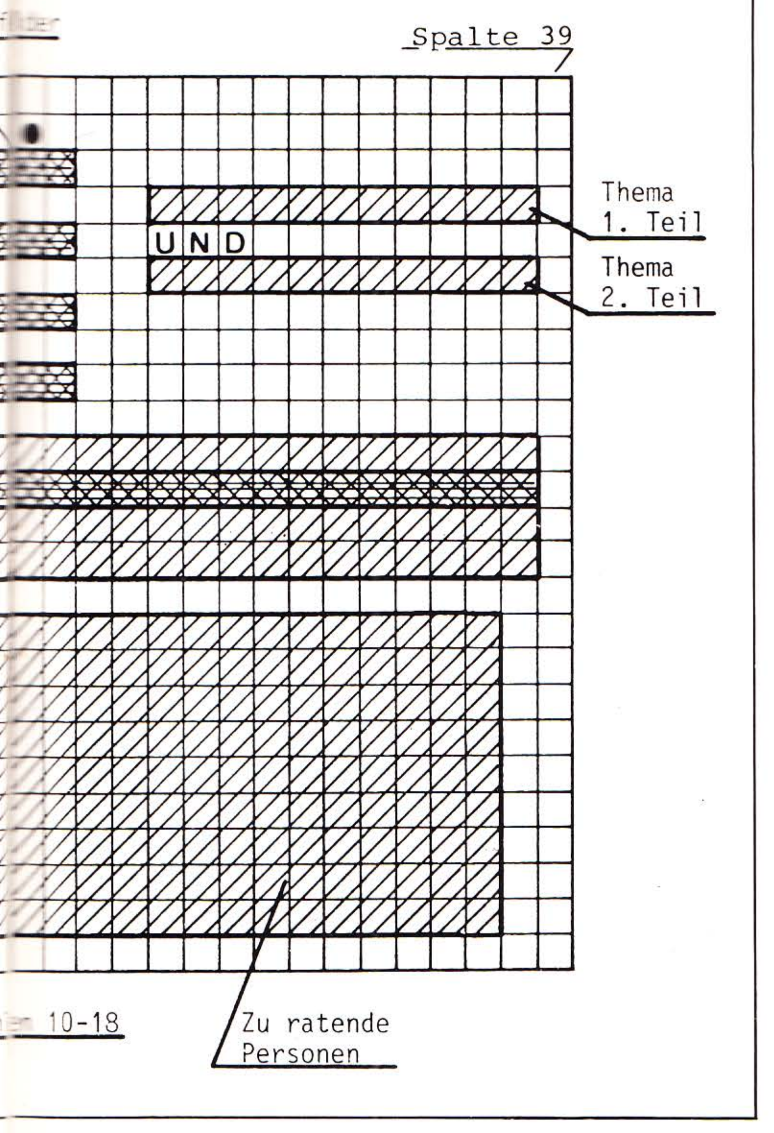
- Hatten Sie schon eigene Spielideen und sahen bisher keine konkrete Möglichkeit, sie zu verwirklichen?
- Sind Sie es leid, nie mit dem Geld auszukommen und hätten Sie gern einen Überblick über Ihre Finanzen?
- Ist es Ihnen schon mal passiert, daß in der Tiefkühltruhe Lebensmittel schlicht und einfach vergessen wurden?
- Brennt es Ihnen auf den Nägeln, zu erfahren, was der geplante Neuwagen im Vergleich zu seiner Konkurrenz tatsächlich im Monat kostet?
- Nerven Sie Ihre Lieben damit, daß Sie jemanden suchen, der Sie nach den Vokabeln abfragt?
- Sind Sie in verschiedenen Schulfächern „schwach auf der Brust", weil Ihnen zum Lernen oft die Motivation fehlt?
- Ernähren Sie sich nach einer Diät und möchten lernen, wie man Nährstoffe und Kalorien abschätzt?
- Fahren Sie oft mit verschiedenen Verkehrsmitteln und möchten wissen, welche der Möglichkeiten für Sie am günstigsten ist?
- Leben Sie in einer Wohngemeinschaft und haben jeden Monat das Problem, die anteiligen Haushaltskosten auszurechnen?
- Haben Sie Lust, mal nur aus Interesse eine private Statistik zu führen?

Falls Sie eine dieser Fragen, die stellvertretend für viele persönliche Bedürfnisse stehen, mit „Ja" beantworten können, sollten Sie ein eigenes BASIC-Programm schreiben.

VOR- UND NACHTEILE VON BASIC

Leider werden die beiden Nachteile des BASIC (Langsamkeit und Unübersichtlichkeit) so oft hervorgehoben, daß man die **Vorteile** leicht übersieht:
1. BASIC ist schnell und leicht für jedermann zu erlernen. Es verlangt durch seine große Nähe zur normalen Umgangssprache nur geringe mathematische und logische Fähigkeiten.
2. BASIC ist nachsichtig und verzeiht dem Programmierer seine Unerfahrenheit und diverse Schlampereien eher als jede andere Programmiersprache.
3. Durch seine Unkompliziertheit gelangt man ungewöhnlich schnell zum programmierten Ziel.
4. Fehler lassen sich bei BASIC rasch und komfortabel aus der Welt schaffen. Wo bei anderen Sprachen eine komplette Umstrukturierung fällig wäre oder nur noch der RESET-Schalter weiterhelfen kann, verwöhnt Sie BASIC mit Fehlermeldungen, die auch meist den Grund für den Fehler angeben. Der C16/Plus4 besitzt als einer der wenigen Heimcomputer sogar spezielle Befehle zur Fehlerbehandlung.

Zu den Nachteilen:
1. Schnelligkeit spielt bei Programmen vor allem dann eine Rolle, wenn auf dem Bildschirm viel bewegt werden soll, wenn's ans Sortieren geht oder wenn sehr viel möglichst schnell ausgerechnet werden soll. Während man im ersten Fall meist die Waffen strecken muß, kommt es beim Sortieren und Rechnen mehr auf die persönlichen Ansprüche und Maßstäbe an.



Bildschirm zum Programm FAMILY-QUIZ

Wenn Sie bereit sind, auch nur geringe Konzessionen in dieser Hinsicht zu machen, läßt sich's mit BASIC gut leben.
2. Ob ein BASIC-Programm chaotisch und für andere unübersichtlich ist, liegt wesentlich mehr am Programmierer als an der Sprache. Vor allem durch die Technik der Unterprogrammierung in Verbindung mit einer Variablenliste kann sich auch ein BASIC-Programm sehr übersichtlich darstellen lassen.
Mit unserer kleinen BASIC-Serie sollen Sie Gelegenheit bekommen, Programmiertechniken, Tricks und Möglichkeiten kennenzulernen, die Sie im Handbuch nicht finden werden.

DAS SOLLEN SIE MITBRINGEN

Zwei Voraussetzungen sollten Sie dafür mitbringen:
1. Sie sollten sich soweit mit BASIC beschäftigt haben, daß Ihnen die wichtigsten Begriffe und Befehle bekannt sind. Das läßt sich an zwei bis drei Abenden nachholen. Ferner wäre es wünschenswert, daß Sie schon einmal ein Programm selbst geschrieben haben.
2. Das Programmieren sollte für Sie mehr Spaß als Arbeit sein. Betrachten Sie Ihren Computer künftig als ein Mittel, Ihrer Kreativität Ausdruck zu geben.
An Hardware-Voraussetzungen genügt der C16/116 in seiner Grundversion (16 KByte), eine Datasette und ein Farbfernseher. Nur wenn Shapes besprochen werden, ist eine Speichererweiterung (bei Plus4 eingebaut) unabdingbar. Das Vorhandensein eines Druckers oder eines Diskettenlaufwerks erweitert die Möglichkeiten.
Wie programmieren Sie in BASIC?
Zunächst sollten Sie auf jeden Fall vermeiden, spontane Ideen ebenso spontan in die Praxis umzusetzen. „Gut Ding will gut Weile haben", bewahrheitet sich auch hier. Manche fixe Idee beim Programmieren zerplatzt wie eine Seifenblase, weil sie sich entweder als undurchführbar, überflüssig oder unlogisch erweist. Erst wenn die Sache nach einigen Tagen Abstand immer noch gut klingt, ist es Zeit, sie zu konkretisieren. Selbst dann ist es nicht ratsam, sich sofort an den Computer zu setzen. Vielmehr sollten Sie sich einen Briefblock schnappen, um einige Dinge schriftlich zu fixieren. Keine Angst, es sollen wirklich keine langen Ausführungen über komplizierte Strukturdiagramme folgen. Sie erleichtern zwar das Programmieren erheblich; erfordern aber Wissen, Logik, Geduld und Übung.

GUT GEPLANT IST HALB PROGRAMMIERT

Es geht auch einfacher. Wir haben uns für diese Artikel Beispielprogramme einfallen lassen, die ohnehin das Abtippen wert sind. In diesem Fall soll das nachfolgende Programm FAMILY-QUIZ dazu dienen, Ihnen meine persönliche Vorgehensweise zur Programmplanung zu demonstrieren. Jeder erfahrene Programmierer wird Ihnen bestätigen, daß ohne Programmplanung oft genug allein der Mißerfolg programmiert wird.
Bei mir vollzieht sich die Planung in drei Stufen:
1. Festhalten der Grundidee,
2. Aufstellen der Spielregeln (oder der Bedienung),
3. Festlegung der Programmstrukturierung.

Zu 1.: Schreiben Sie lediglich knapp und präzise auf, was Sie vom fertigen Programm erwarten.
Beispielprogramm: Quiz für die ganze Familie; mit Fragen, die verwandtschaftsbezogen und personengebunden sind.
Zu 2.: Spielen Sie gedanklich mehrere Möglichkeiten durch, klopfen Sie das Ganze hinsichtlich Praktizierbarkeit und Logik ab und halten Sie den groben Ablauf schriftlich fest.
Beispielprogramm:
- Zwei bis vier Personen können mitspielen.
- Aufteilung der Fragen in drei Themenbereiche:
  - Daten und Jubiläen,
  - Vorlieben und Abneigungen,
  - Begebenheiten und Anekdoten.

Die Themen haben Kennfarben.
- Die Fragen werden so gestellt, daß mit Personennamen geantwortet werden muß, die mittels Kennziffern eingegeben werden.
- Die Themenauswahl erfolgt durch Zufall, die der Fragen ebenfalls.
- Unabhängig von der Fragenbeantwortung wird abwechselnd gespielt.
- Die richtige Beantwortung einer Frage wird mit einem Sternchen auf einem Farbfeld, welches der jeweiligen Themenfarbe entspricht, markiert.
- Sind drei Sternchen auf einem Themenfeld vorhanden, wird der Spieler übersprungen.
- Gewonnen hat der, dem es zuerst gelungen ist, in jedem Themenfeld drei Sternchen zu haben.

Zu 3.: Überlegen Sie, welche Programmschritte für den geplanten Spielablauf voraussichtlich nötig sind. Fertigen Sie in Verbindung mit diesen Überlegungen eine Bildschirmskizze an. Dazu genügt ein karierter Briefblock, auf dem Sie ein Feld von 20*12,5 Zentimetern einrahmen (entspricht 25 Zeilen von 40 Zeichen Länge).
Bei unserem Beispielprogramm sah die Skizze so aus:
Machen Sie sich danach eine Liste der absehbaren Programmschritte und lassen Sie auf der linken Seite einen breiten Rand. Überlegen Sie, welche Reihenfolge die einzelnen Programmschritte haben sollen und schreiben Sie die Startzeilen-Nummern analog zu den Programmschritten an den linken Rand. Um einen schnellen Programmablauf zu erhalten, sollten Sie die selten benutzten Schritte an das Ende des Listings setzen. Heraus kommt eine Liste von Unterprogrammen, die in unserem Fall so aussieht:

9300	Dimensionierung und Zuordnung von Variablen
9200	Titeldarstellung
9100	Anleitung
9000	Abfrage der Spieleranzahl und der Spielernamen
8100	Data der zu ratenden Personen
8000	Spielfeldaufbau
3000	Data der Fragen zum Thema: Daten und Jubiläen
4000	Data der Fragen zum Thema: Vorlieben und Abneigungen
5000	Data der Fragen zum Thema: Begebenheiten und Anekdoten
2300	Fragen auslosen, anzeigen und auswerten
2200	Anzeige des Themas
2100	Auslosen des Themas
2400	Anzeige „Falsch!"
2420	Anzeige „Richtig!"
2800	Löschen der Fragen-Anzeige
2700	Löschen des Lospunktes
2600	Löschen der Themen-Anzeige
6000	Ende

KENNZEICHNEN DURCH REM-ZEILEN

Natürlich kann diese Auflistung nur als Beispiel verstanden werden. Die Aufteilung könnte auch völlig anders aussehen. Nach dieser Arbeit können Sie sich endlich an den Computer setzen und die von Ihnen definierten und zugeordneten Pro-

grammabschnitte mit REMs kennzeichnen. Gehen Sie praktischerweise so vor, daß sich die REM-Zeile eine Nummer vor der Unterprogramm-Startzeile befindet (Beispiel: REM 2399 = Anzeige „Falsch"). Mit dieser geringen Mühe haben Sie sich ein solides Grundgerüst für Ihr Programm geschaffen. Diese Methode hat zwei entscheidende Vorteile:

- Sie können sich durch Unterprogramme quasi eigene Befehle schreiben. GOSUB 2100 ist dann nicht nur ein Sprungbefehl mit einer Adresse, sondern der Befehl „Thema auslosen".
- Die einzelnen Unterprogramme können separat „angetestet" werden, ohne daß auf dem Bildschirm das absolute Chaos ausbricht.

Wurden alle (soweit sinnvoll) Programmabschnitte als Unterprogramme geschrieben, besteht der eigentliche Programmablauf nur noch darin, die Unterprogramme in einer bestimmten Reihenfolge abzurufen. Auch wenn Sie sich nach Jahren das Programm wieder hervorkramen, wird es Ihnen ohne Schwierigkeiten gelingen, sich in den Programmablauf hineinzudenken.

FÜHREN EINER VARIABLENLISTE

Nach diesen Vorbereitungen ergibt sich alles Weitere beim Programmieren. Lediglich eine Mühe sollten Sie noch auf sich nehmen: das Führen einer Variablenliste. Aus ihr soll hervorgehen, welche Variablen verwendet werden, was die jeweilige Variable vertritt (zum Beispiel "X"=Zufallszahl) und eventuell auch die Zeilennummern, in der sie Verwendung findet. Nachfolgend die Variablenliste für unser Beispielprogramm – ohne Zeilennummern, da eine ausführliche Programmbeschreibung nachfolgt:

Dimensionierte Variablen:

CO (3)	=	Kennfarben der Themen
K (3,4)	=	Speicher der Antwort-Sternchen
N$ (4)	=	Spielernamen
T1$ (3)	=	Themenbezeichnungen 1. Teil
T2$ (3)	=	Themenbezeichnungen 2. Teil

Undimensionierte Variablen:

A,B,C,D	=	Universelle Variablen für FOR-NEXT-Schleifen oder als Zähler
KP	=	Horizontale Position des Lospunktes
KX	=	Themen-Zuordnungsnummer
Q	=	Richtige Kennzahl der Antwort
QA	=	Kennzahl-INPUT
S	=	Spielnummer
SP	=	Spieleranzahl
X	=	Zufallszahl
E$	=	Abfrage auf Weiterführung des Spiels
FR$	=	Frage
K$	=	Abfrage auf Korrektheit der Spielernamen
Z$	=	Kette von Leerzeichen

Die Geübteren unter Ihnen können anhand der Variablenliste und der Unterprogrammliste den Programmablauf des Beispielprogramms leicht nachvollziehen. Falls Sie jedoch über wenig Erfahrung in diesen Dingen verfügen, soll hier noch eine ausführliche Erklärung des Programms erfolgen. Sie können das Listing zuerst abtippen und im Nachhinein mit der Beschreibung vergleichen. Effektiver ist es, wenn Sie analog zur Beschreibung das Programm abtippen. In jedem Fall empfiehlt es sich, das Programm abzuspeichern, bevor eigene Einfügungen vorgenommen werden.

Zur Beschreibung:

- In Zeile 1000 wird die Rahmen- und Hintergrundfarbe festgelegt. Durch GOSUB 9300 erfolgt im entsprechenden Unterprogramm die Dimensionierung und Zuweisung der meisten Variablen. A=RND(–TI) in Zeile 9310 dient zur Zufallserzeugung. Das nachfolgende GOSUB 9100 soll die Anleitung abrufen, die Sie später selbst programmieren können.
- Die Zeile 1010 bewirkt durch GOSUB 9000 die Abfragen: Die Abfrage der Spieleranzahl erfolgt in Zeile 9010 (INPUT"";SP). Zeile 9020 schließt eine Über- oder Unterschreitung der Spieleranzahl aus und wiederholt gegebenenfalls die Abfrage (GOTO 9010). Mit Hilfe der Variablen „SP" wird innerhalb der Zeilen 9030 bis 9050 eine Schleife zur Abfrage der Spielernamen gebildet. Ob der eingegebene Spielername (N$) zu lang ist, wird in Zeile 9045 durch LEN (N$(A)) 10 festgestellt. Eventuell wird auch diese Abfrage wiederholt (GOTO 9045). Durch die Zeilen 9060 bis 9090 wird die Gelegenheit gegeben, das Programm fortzuführen (Zeile 9070) oder die gesamte Abfrage zu wiederholen (Zeile 9080: GOTO 9000). Bei unklarer Antwort wird dieselbe Frage erneut gestellt (Zeile 9090:GOTO 9060). Der zweite Befehl in Zeile 1010 (GOSUB 8000) erzeugt den Spielfeldaufbau: Die Zeile 8010 bringt mittels „SP" die Spielernamen (N$) auf den oberen Bildschirmanteil. Die Variable „B" legt hierbei die vertikale Position der Namen fest. Die Zeilen 8020 und 8030 bewirken die Darstellung der themenzugeordneten Farbfelder = Leerzeichen in Zeile 8030. POKE 194,1 in Zeile 8020 ist mit REVERSE ON identisch. GOSUB 2800 in Zeile 8040 erzeugt ein vierzeiliges graues Feld in Bildschirmmitte. Die Namen der zu ratenden Personen werden durch „Q$" in Zeile 8050 eingelesen und zusammen mit der Kennzahl („A") auf dem unteren Teil des Bildschirms dargestellt. Die Zeile 8060 sorgt dafür, daß bei mehr als neun Namen eine zweispaltige Darstellung erfolgt. Ermöglicht wird dies durch Änderung der Variablen „B", die den Zeilenanfang mittels der TAB-Funktion regelt. Das „CHAR 1,19,15" dient lediglich zum Setzen der Ausgangsposition.
- Die Zeilen 2000–2070 bilden eine große Schleife und stellen sozusagen das „Hauptprogramm" dar.
- Für den späteren Programmablauf sorgen die Unterprogramme 2600, 2700 und 2800 in Zeile 2000 dafür, daß alle veränderlichen Anzeigen gelöscht werden.
- Die Zeile 2005 hebt den Namen des betreffenden Spielers revers hervor. GOSUB 2100 führt zur Auslosung des Themas: Hierzu wird innerhalb der Zeilen 2100 bis 2150 eine Schleife gebildet, die erst verlassen werden kann, wenn die Zufallszahl „X" (Zeile 2105) dem Wert 30 entspricht (Zeile 2120); wobei sich „X" zwischen Eins und 50 befinden kann. Solange dies nicht der Fall ist, wird der Lospunkt an drei Positionen fortlaufend abgebildet (Zeile 2110) und gelöscht (Zeile 2130). Die Variable „KX" ändert sich analog zum Lospunkt (Zeile 2140).
- In Zeile 2010 wird geprüft, ob das betreffende

```
10 rem family-quiz============c16 <ih>
11 rem (p) commodore welt team <mj>
12 rem ========================== <pn>
13 rem (c) by peter bergen <fc>
14 rem        hildesheim <la>
15 rem <kn>
16 rem beispielprogramm 1 zur <lh>
17 rem serie "so programmiere ich <ap>
18 rem        in basic" <km>
19 rem <lf>
20 rem basic v3.5 <jb>
21 rem c16/116/p4 <lj>
22 rem ========================== <ih>
1000 color4,2,4:color0,2:gosub9300
:gosub9100 <nm>
1010 gosub9000:gosub8000 <mb>
1020 s=1 <nn>
2000 gosub2600:gosub2700:gosub2800 <ma>
2005 color1,2,4:poke194,1:char1,1,
s*2,n$(s):gosub2100 <gc>
2010 ifk(kx,s)=3thencolor1,3,4:cha
r1,29,4,"besetzt!":fora=1to1500:ne
xt:goto2050 <cn>
2020 gosub2200:gosub2300 <bj>
2030 gosub2600 <hi>
2040 ifk(1,s)=3andk(2,s)=3andk(3,s
)=3thengoto6000 <mm>
2050 color1,2,4:poke194,0:char1,1,
s*2,n$(s) <fm>
2060 s=s+1:ifs>spthens=1 <gl>
2070 goto2000 <dp>
2099 rem auslosen des themas <ba>
2100 color1,1:kp=16:kx=1:poke194,0 <cb>
2105 x=int(rnd(1)*50)+1 <ko>
2110 char1,kp,1,"Q" <co>
2120 ifx=30thensound1,500,5:return <ki>
2130 char1,kp,1," " <gi>
2140 kp=kp+4:kx=kx+1:ifkp=28thenkp
=16:kx=1 <ao>
2150 goto2105 <do>
2199 rem anzeige des themas <bn>
2200 color1,co(kx),4:char1,28,3,t1
$(kx):char1,28,4,"und":char1,28,5,
t2$(kx):return <en>
2299 rem fragen auslosen,-anzeigen
 und  auswerten <ap>
2300 ifkx=1thenrestore3000:x=int(r
nd(1)*#)+1 <hk>
2310 ifkx=2thenrestore4000:x=int(r
nd(1)*#)+1 <op>
2320 ifkx=3thenrestore5000:x=int(r
nd(1)*#)+1 <jd>
2330 a=0 <ld>
2340 do:a=a+1:readfr$:readq <mk>
2350 loopuntila=x <eh>
2360 color1,2,4:poke194,1:char1,0,
11,fr$ <jg>
2370 char1,0,12,"":input"";qa <eo>
2380 ifqa=qthengosub2420:elsegosub
2400 <di>
2390 fora=1to1500:next:return <jj>
2399 rem anzeige falsch! <lm>
2400 gosub2600:poke194,1:char1,30,
4,"falsch!":return <ng>
2419 rem anzeige richtig! <co>
2420 gosub2600:poke194,1:char1,30,
4,"richtig!" <ao>
2425 k(kx,s)=k(kx,s)+1:color1,co(k
x),4:char1,kp-2+k(kx,s),s*2,"*" <pb>
2430 return <jn>
2599 rem loeschen der themen-anzei
ge <lp>
2600 poke194,0:fora=3to5:char1,28,
a," "+b$:next:return <aa>
2699 rem loeschen des lospunktes <ad>
2700 poke194,0:char1,18,1,b4$+b4$:
return <ck>
2799 rem loeschen der fragen-anzei
ge <ei>
2800 char1,0,10,"":color1,2,4:fora
=1to4:poke194,1:printz$:next:retur
n <fe>
2997 rem anzahl der fragen in die
zeilen 2300-2320 eintragen (#) <pd>
2998 rem zu beantwortende fragen (
max.39 zeichen), dahinter kennzahl
 der antwort <oc>
2999 rem daten und jubilaen <am>
3000 data"....",#:rem muster <gb>
3999 rem vorlieben und abneigungen <oe>
4000 data"....",#:rem muster <kb>
4999 rem begebenheiten und anekdot
en <fc>
5000 data"....",#:rem muster <ld>
5999 rem ende <pg>
6000 color1,3,4:poke194,1:char1,29
,4,"gewonnen!" <op>
6010 gosub2800:poke194,1:char1,0,1
1,"":input"noch ein spiel? (ja/nei
n)";e$ <bb>
6020 ifleft$(e$,1)="j"thenfora=1to
3:forb=1to4:k(a,b)=0:next:next:got
o1010 <cn>
6030 end <ml>
7999 rem spielfeldaufbau <me>
8000 scnclr:color1,1 <ab>
8010 b=2:fora=1tosp:char1,1,b,">>>
>>>>>>>>>>>>>>>>>>>":char1,1,b,n$(
a):b=b+2:next <le>
8020 b=2:d=15:fora=1tosp:forc=1to3
:color1,co(c),4:poke194,1 <im>
8030 char1,d,b,b3$:d=d+4:next:d=15
:b=b+2:next <al>
8040 gosub2800 <mi>
8050 print:restore8100:color1,1:b=
0:fora=1to#:readq$:printtab(b)a;"=
```

```
 ";q$                                         <an>
8060 ifa=9thenb=19:char1,19,15,""            <hn>
8070 next:return                              <dj>
8098 rem anzahl der personen in ze
ile 8050  eintragen (#)                       <gh>
8099 rem datas der zu ratenden per
sonen (max.18 / je name max.14 zei
chen)                                         <ea>
8100 data"....":rem muster                    <mg>
8999 rem abfrage der spielerzahl-
und  der spielernamen                         <pc>
9000 gosub9200                                <fb>
9010 print:print:color1,1:input"wi
eviel spieler? (min 2 / max 4)";sp <kp>
9020 ifsp<2orsp>4thensound1,50,20:
goto9010                                      <cf>
9030 fora=1tosp                               <cg>
9040 print:print"name des";a;".spi
elers (max 10 zeichen) ";:inputn$(
a)                                            <am>
9045 iflen(n$(a))>10thensound1,50,
20:goto9040                                   <dm>
9050 next                                     <hm>
9060 print:print:input"alles korre
kt (ja/nein)";k$                              <jn>
9070 ifk$="ja"thenreturn                      <gp>
9080 ifk$="nein"thengoto9000                  <mb>
9090 sound1,50,20:goto9060                    <al>
9099 rem anleitung                            <fl>
9100 gosub9200                                <oj>
9199 rem titeldarstellung                     <nd>
9200 scnclr:color1,3,4:poke194,1:p
rint"*************** family-quiz **
***********":return                           <hj>
9299 rem dimensionierung und zuwei
sung von variablen                            <kc>
9300 vol6:gosub9360                           <lg>
9310 dimn$(4):dimco(3):dimt1$(3):d
imt2$(3):dimk(3,4):a=rnd(-ti)                 <pn>
9320 z$=b$+b$+b$+b5$+b4$                      <pd>
9330 co(1)=3:co(2)=6:co(3)=7                  <gl>
9340 t1$(1)="daten"+b3$+b3$:t2$(1)
="jubilaen"+b3$:t1$(2)="vorlieben"
+b2$                                          <ee>
9350 t2$(2)="abneigungen":t1$(3)="
ereignisse ":t2$(3)="anekdoten"+b2
$:return                                      <md>
9360 b$=chr$(32):b2$=b$+b$                    <oc>
9370 b3$=b2$+b$:b4$=b3$+b$                    <hf>
9380 b5$=b4$+b$:b$=b5$+b5$:return             <ae>
10000 rem ======================              <hc>
10010 rem p r o g r a m m e n d e             <ln>
10020 rem ======================              <if>
```

Thema („KX") schon besetzt ist. Gegebenenfalls erfolgt die entsprechende Meldung, und die Zeilen 2020 bis 2040 werden übersprungen. Die Schleife „FOR A=1 TO 500" dient als Pause.

● Die Zeile 2020 ruft mit GOSUB 2200 die Themen-Anzeige ab und leitet mit GOSUB 2300 zum Fragenteil über: Abhängig von „KX" wird innerhalb der Zeilen 2300 bis 2320 durch RESTORE der zutreffende Fragenteil angewählt.

DATA SPAREN PLATZ

Die jeweils nachfolgende Bestimmung von „X" ermittelt zufällige Zahlenwerte zwischen Eins und der von Ihnen angegebenen Fragenanzahl des betreffenden Themas.
Die Zeilen 2330 bis 2350 bilden eine Schleife, die verlassen wird, wenn der Zählwert „A" mit der Zufallszahl „X" identisch ist. Ein ähnliches Problem wie das in den Zeilen 2100 bis 2150 wird diesmal elegant mit einer DO-LOOP-Schleife gelöst.
Die in dieser Schleife zuletzt eingelesene Variable „FR$" wird durch die Zeile 2360 in Bildschirmmitte als Frage ausgegeben. Das nachfolgende INPUT in Zeile 2370 legt die Antwort in die Variable „QA". „QA" wird in Zeile 2380 mit der richtigen Antwortzahl „Q", eingelesen in Zeile 2340, verglichen.
Das Ergebnis wird durch GOSUB 2400 (Falsch) oder GOSUB 2420 (Richtig) angezeigt. Zudem wird bei richtiger Antwort ein Sternchen in entsprechender Hintergrundfarbe und Plazierung ins Themenfeld gesetzt (Zeile 2425).
● Ob ein Spieler gewonnen hat, klärt die Zeile 2040 durch Abfrage der Konten.
Eventuell verzweigt das Programm zur Endroutine (GOTO 6000): GOSUB 2800 in Zeile 6010 löscht die Bildschirmmitte und fragt über „E$", ob noch ein Spiel folgen soll.
(Der Kniff mit dem „LEFT$" ist nötig, um die reversen Leerzeichen nicht als Antwort gelten zu lassen.)

● Andernfalls bewirkt die Zeile 2050, daß der Spielername wieder dem Hintergrund angeglichen wird.
● Bevor die Schleife 2000 bis 2070 wieder ihren Anfang nimmt, wird in Zeile 2060 der nächste Spieler ermittelt.
Vergleichen Sie abschließend das Eingetippte mit dem Listing und speichern Sie es ab. Behalten Sie das Programm im Computer und beginnen Sie, die nötigen Data und die fehlenden Variablen (Zeilen 2300 bis 2320 sowie Zeile 8050) einzutragen. Speichern Sie auch die vervollständigte Fassung ab.
Wie Sie schon während des Abtippens bemerkt haben, lebt FAMILY-QUIZ von Data. Diese Methode ist die praktische und platzsparende Alternative zu LET. Sie kommt immer dann in Betracht, wenn große Datenmengen, die sich während des Programmablaufs nicht ändern, eingelesen werden sollen. Dieser Vorgang erfolgt zweckmäßigerweise innerhalb einer Schleife (siehe Zeile 2340 bis 2350 und 8050 bis 8070). Setzen Sie immer vor dem Einlesen der Data ein RESTORE, um Verwirrungen zu vermeiden.
Das Listing von FAMILY-QUIZ soll Ihnen lediglich als Grundversion dienen. Sie haben die Möglichkeit, Pausen (zum Beispiel Zeile 2100) und Programmabläufe (zum Beispiel Zeile 2105) zu ändern. Sie können das Programm auch durch Melodien (etwa bei „Falsch!" oder „Richtig!") oder durch ein schönes Titelbild verbessern. Es müßte ohnehin durch eine vernünftige Anleitung (GOSUB 9100 ist noch frei) vervollständigt werden.
Eventuell wäre es auch denkbar, aus FAMILY-QUIZ ein ganz anderes Quiz-Programm zu machen.
Sie sehen: Kreativität ist gefragt! *Peter Bergen* □

COMPUTERN LEICHT GEMACHT

